Итак, перед вами список наиболее часто задаваемых вопросов по Game Maker и GML.

FAQ может и должно постоянно пополняться, так что пишите вопросы, которые на ваш взгляд должны находиться в FAQ. Если я буду с вами солидарен, то я могу сам написать к ним ответ и разместить их в нужной категории. Также, наверняка, во время чтения вы замечаете массу опечаток, так что если вы заметили какую-то ошибку в FAQ, опечатку или что-либо ещё - не стесняйтесь, напишите мне в ЛС, и я всё поправлю.

Жду ваших предложений по улучшению FAQ. Помните, даже незначительные изменения могут упростить использование FAQ.

Авторы, которые приложили руку к созданию этого FAQ:

- Ogion: I.2.12.

Автором всех остальных ответов являюсь я сам, как же и всех примеров, кроме тех, авторы которых обозначены (смотрите раздел "Примеры").

Данную версию (в формате .chm) сделал Dmi7ry (связаться можно через форум, ссылка в правом верхнем углу)
Просьба сообщать о багах, ошибках, опечатках, пожеланиях и т.п.

Старт - в этом подразделе находятся вопросы, которые могут возникнуть у игродела перед началом работы в Game Maker.

Программа - в этом подразделе находятся вопросы, которые могут возникнуть у игродела при работе непосредственно с Game Maker'ом.

Игрострой - в этом разделе находится теоретическая информация о Game Maker'е.

1.1.1. Что такое Game Maker?

Ответ: Game Maker — один из самых известных конструкторов игр. Это не движок. Первую версию GM Разработал Марк Овермарс в 1999 году, однако, начиная с версии 7.0 разработкой GM занимается команда YoYo Games. Официальный сайт конструктора на данный момент - yoyogames.com. Скачать Game Maker для Windows и Mac можно здесь: http://www.yoyogames.com/gamemaker/try

1.1.2. Что такое GML?

Ответ: Game Maker Language (GML) — это интерпретируемый язык программирования, встроенный в Game Maker. Он предлагает значительно большую функциональность, чем встроенные действия lib библиотек GM.

1.1.3. Какая последняя версия Game Maker?

Ответ: На данный момент, последняя версия GM - 8.1. Впрочем, постоянно выходят обновления, исправляющие баги текущей версии и добавляющие новые возможности.

1.1.4. На каких платформах работают GM-игры?

Ответ: На данный момент существует версия Game Maker для Windows и Mac. У YoYo Games так же имеется конвертер для портирования GM игр на PSP, iOS (iPhone, iPad), Android и HTML5. Так же, в планах у YoYo Games выпустить продукт под названием "GameMaker Studio", некую особую версию GameMaker, включающую возможность портирования игр на различные платформы.

1.1.5. Что такое Pro и Lite версии Game Maker? Какая стоимость Game Maker?

Ответ: Начиная с GameMaker версии 8.1, обычная Pro версия стала называться Standart (а Lite сохранила свое название). До первого июня 2011 года GameMaker Standart можно купить за $25, однако после этого стоимость GameMaker повыситься до $39.99. Покупка GameMaker 8.x будет действительна для всех версий 8.x

Информация из русифицированного справочного руководства к **Game Maker 8.0**:

▫ Читать

Game Maker предоставляется в двух версиях, Lite Edition (облегченная редакция) и Pro Edition (профессиональная редакция).

Lite Edition подходит для тех, кто только делает первые шаги в разработке игр. Она

совершенно бесплатна, но имеет ограниченную функциональность. Также при запуске игр показывается логотип, а при использовании программы будет постоянно выходить напоминание о возможности обновления. Если Вы собираетесь пользоваться Game Maker регулярно, Вам настойчиво рекомендуется обновить программу до Pro Edition.

В Pro Edition значительно расширена функциональность и не показываются логотипы и сообщения. Конкретно в Pro Edition доступны следующие возможности:

- Не показывается логотип Game Maker при запуске игр.
- Не показываются всплывающие сообщения о возможности обновления.
- Возможность отрисовывать повернутые, окрашенные и прозрачные спрайты.
- Дополнительные опции в редакторах спрайтов и изображений.
- Дополнительные действия, такие как проигрывание CD музыки, рисование повернутого текста и окрашенных форм.
- Использование звуковых эффектов и позиционирования источников звука.
- Использование загрузочных экранов с фильмами, изображениями, веб страницами, текстом и т. д.
- Система частиц для эффектов взрыва, салюта, пламени, снега, дождя и других.
- Дополнительные продвинутые функции рисования, например, для отрисовки окрашенного текста и текстурированных полигонов.
- Возможность создание 3D игр, при использовании функций для 3D графики.
- Возможность создания многопользовательских проектов для игры через интернет или локальную сеть.
- Возможность задавать свои собственные эффекты перехода между комнат.
- Вы можете использовать функции для создания, загрузки и изменения ресурсов (спрайтов, фонов и т.д.) "на лету", прямо во время игры.
- Доступны функции для работы с различными структурами данных.
- Функции для планирования движения.
- Возможность вложения файлов в исполняемые файлы игр, которые можно использовать во время запуска игр.
- Функциональные возможности можно улучшать пакетами расширений. Их может сделать любой человек и распространять, как правило, совершенно бесплатно.
- Три таких пакета расширения уже включены в дистрибутив, они добавляют новые эффекты перехода между комнатами, диалоги windows и функции для принтера.
- Вы можете определять свои собственные события триггера.
- Вы можете импортировать и экспортировать ресурсы, что поможет Вам легко совмещать ваши проекты.

Обновление с версии Lite Edition до Pro Edition стоит **20 Евро** или **US $25**. Этот взнос делается один раз и будет действительным, как минимум, для всех версий Game Maker 8.x.

Информация из официального справочного руководства к Game Maker 8.1.69:

Читать

GameMaker comes in two editions, the Lite Edition and Standard Edition.

The **Lite Edition** is meant for those that take their first steps on the path of developing games. It can be used for free but is limited in its functionality. Also it shows a popup logo when running games and will regularly remind you of upgrading the program. When you are using GameMaker regularly you are strongly recommended to upgrade from the Lite Edition.

Standard Edition contains considerably more functionality and does not display any logos or popup messages. More precisely, Standard Edition has the following additional functionality:

- No GameMaker logo is shown when running a game.
- No GameMaker TV logo in the corner of a game.
- No GameMaker advert screen on exit.
- No regular popups remind you of upgrading.
- You can use color blended sprites, which can be used for many special effects and easy shadows.
- There are additional options in the sprite and image editors.
- There are additional actions for e.g. CD music, rotated text, and colorized shapes.
- You can use special sound effects and positional sound.
- You can create splash screens with movies, images, webpages, texts, etc.
- There is a particle system to create explosions, fireworks, flames, rain, and other effects.
- A number of advanced drawing functions are available, for example colorized text and textured polygons.
- It is possible to create 3D games using functions for 3D graphics.
- It is possible to create multiplayer games that can be played over a network.
- You can define your own room transitions.
- You can use functions to create, load, and modify resources (sprites, backgrounds, etc.) while the game is running.
- There is a collection of functions to create and use data structures.
- There are functions for motion planning.
- You get the possibility to include additional files in the game executables that can be used when the game is run.

- Standard Edition can be easily extended using extension package. These can be made by everybody and will in general be provided free of
- charge.
- Three such extension packages are included adding many room transitions, windows dialogs, and printing facilities.
- You can define your own trigger events.
- You can export and import resources, which makes it easier to collaborate on games.

Upgrading from Lite Edition costs only $39.99 (subject to change). This is a one-time fee that will be valid for all versions 8.x of GameMaker.

When you are running the Lite Edition, you can upgrade by going to the Help menu, and selecting **Upgrade from Lite Edition**.

If you purchased GameMaker before (and hence, have an license key), you can go to the Help menu, and pick **Enter License Key**. Once you have entered a valid, your copy of GameMaker will be upgraded. You must be connected to the internet for the upgrade to work.

While GameMaker itself does not require an internet connection, it will require occasional access to maintain the license. If you can not connect your computer to the internet, you can download the license check file from YoYo Games website, and point the auto update system to it. if you fail to provide a valid file, or an internet connection when requested, your copy of GameMaker will be downgraded to Lite until such time as you do.

1.1.6. Можно ли создавать на Game Maker программы?

Ответ: Да. Достаточно посмотреть коснтруктор игр Noobster, HTML редактор HyperPage, браузер GMB, музыкальный проигрыватель visual music и торрент-клиент GMTorrent

1.1.7. Можно ли создавать на Game Maker 3D игры?

Ответ: Безусловно. Взгляните на гоночные симуляторы SWERVE и Park Racer Game, GTA-подобные игры Total Anarchy и Crimelife 3, онлайн шутер Metroid online, оффлайн шутер Ice Arena 3d, клон «портала» - Power Gates, авиасимулятор Aces High Over verlor Island и вестерн Guns and Spurs!

1.1.8. Можно ли создавать на Game Maker онлайн игры?

Ответ: Да. Можете сыграть в Arcane Adventures, Almora Online и Gang Garrison 2.

1.1.9. Где найти русское справочное руководство к Game Maker 8?

Ответ: Вы можете скачать его здесь:
http://gmakers.ru/index.php?action=tpmod;dl=get220
Или прочитать онлайн здесь:
http://gmakers.ru/gamemaker_help/

1.2.1. У меня появились / я скачал файлы с расширением *.gb1 - *.gb9. Что это?

Ответ: Это бэкап-файлы игр, сделанных на GM. Каждый раз когда вы сохраняетесь - бэкап файлы обновляются. В случае, если сохранили игру с критической ошибкой - вы можете удалить текущий исходник и работать с бэкап файлом. Для этого просто откройте его с помощью GM (ПКМ -> Открыть с помощью -> Game Maker 8). Выбрав пункт **Preferences** в меню **File** - вы увидите окно настроек GM, где во вкладке General вы можете либо вовсе отключить создание бэкапов, убрав флажок с **«Keep backup copies of files»**, либо изменить их количество, изменив значение **«Maximum number of backups»**.

1.2.2. Текст в редакторе кода стал меньше / больше. Как восстановить нормальный размер?

Ответ: Используйте функциональные клавиши **F7** и **F8** (только для Game Maker 8).

1.2.3. Меню помощи в редакторе кода перестало появляться. Что делать?

Ответ: Нажмите **F11** (только для Game Maker 8).

1.2.4. Что такое *.lib файлы? Я скачал lib файл, как мне его установить в Game Maker?

Ответ: lib файлы - библиотеки действий GM. То есть, файлы, содержащие в себе описания "квадратиков", "пиктограмм" - называйте как хотите. Изначально в Game Maker их 7: **move**, **main1**, **main2**, **control**, **score**, **extra** и **draw**. Чтобы установить *.lib файл - достаточно положить его в папку lib, которая находится в директории Game Maker (к примеру: "C:\Program Files\Game_Maker8\lib") и перезапустить GM, чтобы изменения вступили в силу. Чтобы деинсталлировать lib-библиотеку, просто удалите файл из папки.

1.2.5. Могу ли я создавать свои *.lib файлы?

Ответ: Да. Для этого используйте программу **Library Maker**, которую можно скачать по ссылке: http://www.yoyogames.com/downloads/extensions/extmaker.zip

1.2.6. Я скачал файлы с расширением *.gex, что это?

Ответ: Это пакеты расширений, возможность использования которых была добавлена в Game Maker 7. После их установки в ваш проект добавляются новые функции и/или lib-библиотеки.

1.2.7. Как использовать функции пакета расширения?

Ответ: Перед тем как добавить пакет расширения в ваш проект - вам нужно установить его в Game Maker. Выберете из меню **Resources** пункт **Select Extension Packages** (Shift+Ctrl+E). Перед вами появится окно, в котором вы можете добавить какой-то пакет расширения в вашу игру или просмотреть информацию о нём и прочитать справочное руководство. Жмите кнопку "**Install**". В открывшемся окне вы можете удалить из Game Maker выбранный пакет расширения, нажав кнопку "**Uninstall**" и установить новый, нажав кнопку "**Install**" и выбрав ***.gex** файл. После установки пакетов расширений вы можете добавить какие-то из них себе в проект. Для этого вернитесь в первое окно (нажмите "OK"), выберите имя пакета в списке справа, и нажмите на появившуюся кнопку между списками. Убрать пакет расширения из вашего проекта вы можете аналогичным образом.
Функции пакетов расширений в данном проекте можно посмотреть, выбрав из меню **Scripts** пункт **Show Extension functions**.
Подробнее здесь: http://gmakers.ru/gamemaker_help/source/files/215_00_extensions.php

1.2.8. Могу ли я создавать свои *.gex файлы?

Ответ: Да. Для этого используйте программу **Extension Maker**, которую можно скачать по ссылке: http://www.yoyogames.com/downloads/extensions/extmaker.zip

1.2.9. Сохраняются ли шрифты в GMK файле?

Ответ: Нет. Если в исходнике используется шрифт, которого нет на данном компьютере - в игре будет использован **Arial**. Исполняемый файл игры (***.exe**) напротив, хранит все шрифты в себе.

1.2.10. Как защитить exe от декомпиляции?

Ответ: Можно использовать антидекомпилятор: http://gmc.yoyogames.com/index.php?showtopic=422511 (работает с играми созданными на GM8.0, GM7.0 и GM6.1 (предварительно конвертированными для запуска на Windows Vista), а так же программу MoleBox.

1.2.11. Как изменить меню, которое появляется при нажатии на F2 в редакторе кода?

Ответ: К папке с установленным Game Maker 8 вы можете найти файл "**snippets.txt**" (по умолчанию: C:\Program Files\Game_Maker8\snippets.txt). Теперь вы можете создать свои собственные шаблоны кода - это бывает очень полезно.

1.2.12.У меня Windows Vista/7, и любая, даже содержащая пустую комнату игра грузится очень долго, подолгу застревая на "Preparing Sounds". Как это исправить?

Ответ: Откройте Диспетчер Устройств и из списка аудиоустройств удалите все, кроме последнего. Возможно, это придется делать после каждого выключения компьютера.

1.3.1: Что такое шаг, что такое скорость комнаты?

Ответ: В Game Maker за секунду проходит определённое количество шагов. Это количест
можно задать в свойствах комнаты: **settings -> Speed**). Событие **step event** называется соб
выполняется каждый шаг. К примеру, если в событии **step event** объекта вы напишете код
комнаты на 30 шагов, то за секунду переменная **score** увеличится на 30. Таким же образом
за секунду у спрайта сменится 30 кадров, потому что переменная **image_speed** хранит в се
спрайта каждый шаг. То же относится и ко времени. Если вы напишете «alarm[0] = 1», то
секунды (для общего случая - 1/room_speed).

1.3.2. Что такое FPS? Это тоже самое, что и скорость комнаты?

Ответ: В идеале, за секунду должно выполнится <room_speed> шагов, но иногда скорость
В этом случае время на выполнение шага превышает «1000/room_speed» миллисекунд (сек
должно выполниться <room_speed> шагов). Количество шагов, которые выполняются за се
Per Second – количество кадров в секунду, **не путать с First-Person Shooter**). Например, ес
23, то в среднем шаг выполняется 43.4 миллисекунды(1000/23). В идеале, он должен выпо
Не забывайте, что в событие шага входит не только те действия, которые вы вписали в step
экране, обработка событий и др. Если fps меньше чем **room_speed** - значит нужно оптими
или часто вызываемых событиях (особенно стоит обратить внимание на **step**, **end step**, **beg
внимание, что fps обновляется только раз в секунду. FPS можно отобразить в заголовке ок
либо объекта:

Код:

```
room_caption = 'fps: ' + string(fps) + ' / ' + string(room_speed);
```

1.3.3. Как работают функции рисования в GM? Что делают функции screen_redraw(), sc

Ответ: Для начала следует понять, что вызывая функции рисования - вы рисуете не прямо
начале события **draw** внутренний буфер очищается, а в конце - отображается на экран. Ри
рисуете на внутренний буфер, но он не отображается на экран. Функция **screen_refresh** пр
изображение на экране - она отображает внутренний буфер. Функция **screen_redraw** снач
заполняет внутренний буфер цветом фона, рисует фоны, вызывает события рисования всех
изображение на экране (отображает внутренний буфер). Механизм использования глубин
начале события **draw event** строится приоритетная очередь, в которой роль значения игра
глубина. Далее вся графика отображает в порядке приоритета, именно потому глубину ри
приоритетная очередь уже построена. В случае с 3D, используется буфер глубины, и там с
Безусловно, мы не можем знать наверняка, как всё устроено в Game Maker'e, если только с
расскажут. Это не точное описание внутренних механизмов GM, это лишь общий принци
факторы указывают на то, что он именно такой.

1.3.4. Что такое комната? Что такое persistent комнаты, game save и game load?

Ответ: Давайте поразмышляем. Что такое комната? Комната - это набор определённых на
объектов и тайлов. На самом деле, самая первая, "нулевая" комната - это комната загрузки
внутренние ресурсы , инициализируется окно игры и загружается первая игровая комната.
подразумеваю следующие действия:

1) Назначаются значения всем переменным комнаты:
   1.1) Индекс комнаты - **room**
   1.2) Заголовок комнаты - **room_caption**
   1.3) Ширина и высота комнаты - **room_width** и **room_height**
   1.4) Скорость комнаты - **room_speed**
   1.5) Постоянность комнаты - **room_persistent**
   1.6) Цвет фона и его видимость - **background_color** и **background_showcolor**
2) Загружается массив видов:
   2.1) Видимость в момент начала комнаты - **view_visible[0..7]**
   2.2) Позиция, ширина и высота вида: **view_xview[0..7], view_yview[0..7], view_wview[0..7**
   2.3) Позиция, ширина и высота порта: **view_xport[0..7], view_yport[0..7], view_wport[0..7**
   2.4) Граница вокруг преследуемого объекта: **view_hborder[0..7] и view_vborder[0..7]**
   2.5) Вертикальная и горизонтальная скорость: **view_hspeed[0..7] и view_vspeed[0..7]**
   2.6) Преследуемый объект - **view_object[0..7]**
   2.7) Поворот вида - **view_angle[0..7]**
3) Загружается массив фонов:
   3.1) Видимость в момент начала комнаты - **background_visible[0..7]**
   3.2) Отображать ли фон над объектами - **background_foreground[0..7]**
   3.3) Индекс фонового изображения - **background_index[0..7]**
   3.4) Позиция - **background_x[0..7]** и **background_y[0..7]**
   3.5) Замощение комнаты по горизонтали/вертикали - **background_htiled[0..7]** и **backgrou**
   3.6) Вертикальная и горизонтальная скорость - **background_hspeed[0..7]** и **background_vs**
   3.7) Масштаб фона - **background_xscale[0..7]** и **background_yscale[0..7]**
   3.8) Цвет смешивания для фона - **background_blend[0..7]**
   3.9) Непрозрачность фона - **background_alpha[0..7]**
4) Создаются все тайлы:
   4.1) Позиция тайла - **x**, **y**
   4.2) Индекс фона - **background**
   4.3) Часть фона - **left, top, width, height**
   4.4) Глубина тайла
   4.5) **id** тайла, начинается с 10 000 001 (включительно)
5) Создаются все объекты, вычисляется **instance_count** и заполняется массив **instance_id**:
   5.1) Позиция - **x, y**
   5.2) Индекс объекта - **object_index**.
   5.3) **id**, начинается с 100 001 (включительно)
6) Выполняется **creation code** объектов (который задаётся в комнате)

7) Выполняется **creation event** объектов
8) **Game start** (Выполняется только в момент начала игры)
9) **Room creation code** (выполняется код, задаваемый в настройках комнаты)
10) **Room start event**

Действия, не обозначенные символами, выполняются в любом случае - при загрузке самой сохранения, при загрузке persistent комнаты и т.д. Действия, обозначенные так - выполняю комнату, а так же для всех не persistent комнат. Если вы вышли из постоянной комнаты, а выполнятся. Действия, обозначенные так - выполняются только при сохранении/загрузке постоянной комнаты. Эти настройки нельзя изменить в редакторе комнат, потому, при пе загружаются, а получают значения по умолчанию. Действие же обозначенное так - событи раз, при первой загрузке самой первой комнаты. Обратите внимание, что вне зависимости - событие Room start выполняется всегда.

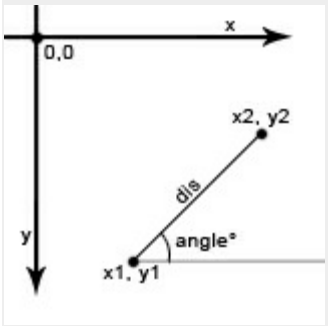
Этот список действий может внести некоторую ясность в то, что конкретно происходит в сохранения постоянных комнат.

Если установлен persistent, при переходе в другую комнату - все данные о текущем состоян видов, фонов, списки объектов и тайлов и т.п., а так же все локальные переменные всех об сохраняются, динамические структуры данных - нет). При сохранении игры сохраняются д включая текущую (вне зависимости, постоянная она или нет) + глобальные переменные и объекты, загруженные спрайты, фоны, звуки, изменения в действиях объектах (object_even позиции проигрывания звуков, настройки игры (полноэкранность и т.п.), частицы - всё эт механизмом сохранения GM. Что же происходит при загрузке? Всего лишь заменяются да текущую (не зависимо, постоянная ли она). Я лично предполагаю, что все значения локал **code** объектов, и этот код выполняется всегда. Однако, я всё равно обозначил его цветом,

Написать свой механизм сохранения проще, чем кажется, стоит только всё продумать.

1.3.5. Что такое lengthdir_x и lengthdir_y и как им пользоваться?

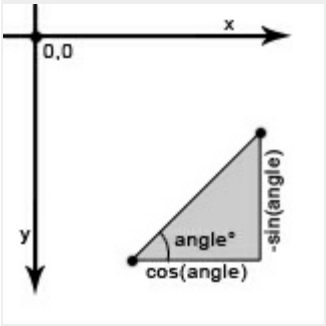
Ответ: Чтобы вам легче было разобраться - начнём с небольшой задачи. Есть позиция, (x1 dis. Как вычислить позицию (x2,y2), расположенную на расстоянии dis от позиции (x1,y1)



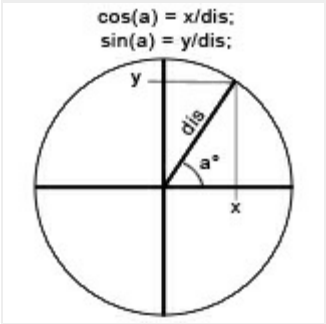
Сейчас немного теории.
Прямоугольный треугольник.
Прямоугольный треугольник - это треугольник с углом 90° (прямым углом). Катет - одна и треугольника, образующая прямой угол. Гипотенуза - сторона прямоугольного треугольни

Синус и Косинус.
Из школьного курса геометрии вам должно быть известно два определения синуса:
1) синус острого угла это отношение катета, лежащего напротив этого угла к гипотеузе.
2) синус это ордината (y-координата, если кто забыл) точки, лежащей на единичной окруж
Косинус так же имеет два общеизвестных определения:
1) косинус острого угла это отношение катета, выходящего из этого угла (прилежащего ка
2) косинус это абсцисса точки, лежащей на единичной окружности.
Единичная окружность - это окружность с радиусом 1.

Так как косинус (cos(angle)) по определению - (x2-x1)/dis, стаёт очевидно, что x2-x1 мы мо
на dis: cos(angle)*dis, а y2-y1 с помощью синуса: sin(angle)*dis. Однако, в декартовой систе
а не вниз, как в Game Maker. Потому, следует отразить координату y, отнять её от нуля или



Но что, если угол angle не острый и не удаётся построить прямоугольный треугольник как



т.е. в любом случае cos и sin будут работать как надо.

Итак, формулы верны. Что же такое **lengthdir_x** и **lengthdir_y**?

Код:

```
lengthdir_x(len, angle) = cos(angle)*dis;
lengthdir_y(len, angle) = -sin(angle)*dis;
```

Подведём итог:

Код:

```
x2 = x1 + lengthdir_x(dis, angle);
y2 = y1 + lengthdir_y(dis, angle);
```

1.3.6. Что за переменная image_single, которой нет в справке?

Ответ: Эту переменную оставили для совместимости с играми, разрабатываемыми в стар

кадр и автоматически устанавливает скорость анимации на 0. К примеру, вместо

Код:

```
image_single = 3;
```

Можно написать:

Код:

```
image_index = 3;
image_speed = 0;
```

Не смотря на её удобность в некоторых случаях - рекомендуется использовать **image_inde**

1.3.7. Что это за функции, которые начинаются с приставки action_ и которые не описа

Ответ: Это встроенные функции в Game Maker, точно повторяющие действия стандартны в справочном руководстве и для них не показывается подсказка в редакторе кода. Их реко которые только-только перешли на код. В этом документе можно увидеть полный список http://www.blackratstudios.com/games/DD_to_GML_7/Drag%20and%20Drop%20Icons%20and

1.3.8. Как работает оператор mod?

Ответ: Оператор **mod** - это деление по модулю. Операция деление по модулю это нахожд другое, т.е. той части числа, которая не поделилась нацело. К примеру, если поделить чис от деления будет 2 - число 90 можно записать как 11*8 + 2:

Код:

```
90 / 11 = 8.18181818...
90 div 11 = 8
90 mod 11 = 2
90 = (90 div 11) * 11 + (90 mod 11)
```

1.3.9. Зачем нужен оператор var?

Ответ: Когда вы создаёте объект - вы автоматически создаёте 50 локальных переменных

Посмотреть

id
object_index
persistent
solid
depth
mask_index
visible

xstart

```
ystart
xprevious
yprevious
x
y

hspeed
vspeed
speed
direction
friction
gravity
gravity_direction

sprite_index
sprite_width
sprite_height
sprite_xoffset
sprite_yoffset

image_index
image_xscale
image_yscale
image_angle
image_blend
image_alpha

image_number
image_speed
image_single

bbox_left
bbox_right
bbox_bottom
bbox_top

path_endaction
path_index
path_orientation
path_position
path_positionprevious
path_scale
path_speed

timeline_index
timeline_loop
timeline_position
timeline_running
timeline_speed

alarm
```

Заданные локальные переменные в этом объекте остаются с ним до удаления или до конц
переменные в скрипте, а потом вызываете этот скрипт в объекте - все новые переменные
собой память. Оператор var предназначен для объявления временных переменных, которы
выполнения этого скрипта. Эти переменные не являются локальными для данного экземп

примеру, такой код не выдаст ошибки:

Код:

```
var i; // Объявляем временную переменную
i = 100;
with o_test // Обращаемся к другому объекту
  a += i; // Используем временную переменную в этом объекте
```

Создавать временные переменные можно не только в скриптах, но и в действиях объектов

1.3.10. Когда следует использовать switch, а когда if?

Ответ: Оператор **switch** следует использовать в том случае, когда количество вариантов зн раза. Конечно, многое зависит от ситуации, иногда даже четыре проверки удобно записать оператора **switch**:

Код:

```
switch varible
{
  case value0: action1(); break;
  case value1: action2(); break;
  case value2: action3(); break;
  case value3: action4(); break;
  default: action5();
}
```

Оператор **if**:

Код:

```
if varible == value0 then
  action1()
else if varible == value1 then
  action2()
else if varible == value2 then
  action3()
else if varible == value3 then
  action4()
else action5();
```

Практика

Базовая работа с кодом и простые решения - очень простые вопросы.
Движение и столкновения - все вопросы, касающиеся перемещения чего-либо и проверки столкновений.
Рисование - все вопросы, касающиеся отображения графики и данных.
Другое - все вопросы, которые не подходят под тематику остальных подразделов.
Оптимизация - все вопросы, связанные с оптимизацией.
ИИ - все вопросы, связанные с написанием искусственного интеллекта.

2.1.1. Как ограничить значение переменной?

Ответ: Для начала, следует учесть, что вы можете получить небольшой, но всё же выигрыш в производительности, проверяя значение переменной исключительно фактически после её изменения. То есть, если жизни игрока уменьшаются при столкновении с монстром и с огнём - немного производительней будет поставить две проверки в эти события, чем писать одну в **step event**, которая будет выполнятся каждый шаг. Однако, не стоит злоупотреблять этим в ущерб сопровождаемости кода. К примеру, если вам понадобится как-то изменить событие смерти игрока - вам придётся менять его в двух событиях (в лучшем случае), а не в одном (в случае со step event). Чтобы этого не происходило, лучше выносите повторяющиеся куски кода в скрипты или **user events**. Последние можно выполнить с помощью команды **event_user**(номер_события).

Итак, при увеличении или уменьшении вы можете использовать такие простые проверки:

Код:

```
variable += x;
if variable > variable_max // Если переменная больше указанного значения, то...
  variable = variable_max; // Устанавливаем переменной максимальное значение

variable -= x;
if variable < variable_min
  variable = variable_min;
```

Ещё один вариант решения через **if**:

Код:

```
if (variable>variable_max) {variable=variable_max} else {variable+=x}
```

Почти то же самое, но здесь переменная изменяется после проверки, а не до неё.

Слегка понизив читабельность можно добится большей компактности кода, используя функции **min** и **max**:

Код:

```
variable = min(variable + x, variable_max); // Если variable_min + x больше variable_max - выбираем меньшее, то есть, variable_max.
variable = max(variable - x, variable_min);
```

Часто в различных примерах вы можете встретить такую запись:

Код:

```
variable = max(variable_min, min(variable, variable_max));
```

Однако, есть способ лучше, который я и предпочитаю использовать:

Код:

```
variable = median(variable_min, variable, variable_max); // Находим среднее из значений, если variable < variable_min - средним становится variable_min и аналогично для variable_max.
```

Например, если вы хотите ограничить передвижение игрока некоторыми рамками, скажем, сделать, чтобы он не выходил за границы комнаты - вы можете написать так:

Код:

```
x = median(0, x, room_width - sprite_width);
y = median(0, y, room_height - sprite_height);
```

2.1.2. Мне нужно написать текст с указанием значения переменной, как?

Ответ: Пропишите отдельному объекту в событие draw, такой код:

Код:

```
draw_text (x, y, ' Текст до переменной ' + string(переменная) + ' Текст после');
```

Не забудьте, что переменная может быть названа только на английском языке, без пробелов и начинаться она должна с буквы.
Так же, если переменная всегда хранит текстовое значение - её не обязательно заносить в string().

2.1.3. Как сделать, чтобы герой TDS смотрел на мышь?

Ответ: Так как направление 0 - вправо, нужно повернуть спрайт игрока вправо и прописать такой код в step event объекту:

Код:

```
image_angle = point_direction (x, y, mouse_x, mouse_y);
```

Так же убедитесь, что в draw event вы не рисуете спрайт объекта без использования переменной image_angle. К примеру, если вы рисуете спрайт таким образом:

Код:

```
draw_sprite(sprite_index, image_index, x, y);
```

следует заменить эту строку на:

Код:

```
draw_sprite_ext(sprite_index, image_index, x, y, image_xscale, image_yscale, image_angle, image_blend, image_alpha);
```

2.1.4. Как запросить имя игрока, и в последствии использовать его в сообщениях и диалогах?

Ответ: Запросить имя можно таким образом:

Код:

```
global.Hero_name = get_string('Как вас зовут?', 'DeatHSoul');
```

После этого, введённое игроком имя будет хранится в глобальной переменной **Hero_name**, которая будет сохранять свое значение во всех комнатах, и вы можете использовать её, чтобы получить имя игрока. Например, так:

Код:

```
show_message('Привет ' + global.Hero_name + '!');
```

2.1.5. Как программно отразить спрайт по вертикали или горизонтали?

Ответ: Чтобы отразить спрайт по горизонтали - нужно параметрам **image_xscale** и **image_yscale** присвоить значение **-1**. Однако, убедитесь, что ваш спрайт рисуется с использованием переменных **image_xscale** и **image_yscale**. т.е. если вы рисуете спрайт функцией **draw_sprite,** замените её на функцию **draw_sprite_ext**.

2.1.6. Как повернуть комнату?

Ответ: Используйте **view_angle[0..7]**, и убедитесь, что у вас включен указанный вид. Если вы хотите, чтобы вид поворачивался вместе с игроком - учтите, что его нужно поворачивать в противоположную сторону. Вместо:

Код:

```
view_angle[0] = direction;
```

пишите:

Код:

```
view_angle[0] = -direction;
```

2.1.7. Как сделать паузу между выстрелами?

Ответ:
[create event]:

Код:

```
shooting_ready = 1; // Указываем, что игрок готов стрелять
```

step event:

Код:

```
if (shooting_ready) // Если игрок готов стрелять
{
 // ...
```

```
 // Стреляем
 alarm[0] = room_speed*<количество_секунд>; // Устанавливаем время, через которое игрок снова будет готов стрелять
 shooting_ready = 0; // Указываем, что игрок более не может стрелять
}
```
alarm 0 event:
Код:
```
shooting_ready = 1; // После истечения времени указываем, что игрок готов стрелять
```

2.1.8. Как получить имя объекта по его id?

Ответ: name = **object_get_name**(ID.object_index);
Где ID - **id** объекта, имя которого требуется получить.

2.1.9. Как получить индекс объекта по его имени (строка)?

Ответ: object = **execute_string**('return '+**string**(obj_name));

2.1.10. Как сделать простую паузу одной кнопкой?

Ответ: Один из самых простых алгоритмов состоит в использовании функции **keyboard_wait** (код нужно вставить в событие нажатия соответствующей клавиши):
Код:
```
/// Отрисовка сообщения о паузе игры
draw_set_color(c_black);
draw_set_alpha(0.3);
draw_rectangle(0, 0, view_wview, view_hview, 0); // Отрисовка фона. Если у вас не используются виды - замените view_wview и view_hview на room_width и room_height соответственно.
draw_set_alpha(1);
draw_set_halign(fa_center);
draw_set_valign(fa_middle);
draw_text(view_xview + view_wview/2, view_yview + view_hview/2, 'PAUSE'); // Отрисовка текста. Если у вас не рисует русский текст - посмотрите вопрос 2.4.6.

// Обновляем экран
screen_refresh(); // Эта функция в действительности отображает на экране всё то, что мы нарисовали ранее

// Останавливаем игру до нажатия какой-либо клавиши
keyboard_wait();
```
Если вы хотите, чтобы игра продолжалась только при нажатии какой-то конкретной клавиши, к примеру, клавиши P (или, скажем, клике мыши) - замените последнюю строку скрипта (keyboard_wait()) на:
Код:
```
io_clear(); // Очищаем состояние клавиатуры

while true // Запускаем вечный цикл
{
  if keyboard_check_pressed(ord('P')) break; // Если игрок нажал P - выходим из цикла
  keyboard_wait(); // Ожидаем нажатия какой-нибудь клавиши
}

io_clear(); // Снова очищаем состояние клавиатуры, чтобы пауза не влючилась сразу же после выключения
```

2.1.11. Как сделать, чтобы значение переменной сохранялось между комнатами?

Ответ: Первый способ - сделать переменную глобальной.Пример объявления глобальных переменных:

Код:

```
global.variable = 100;
globalvar variable2;
variable2 = 100;
```

Второй способ - сделать объект, в котором объявляется переменная - постоянным (persistent). Все глобальные переменные можно посмотреть в дебаг режиме: Tools -> Show Global Variables.

2.1.12. Как определить, на какую кнопку в сообщении нажал пользователь?

Ответ:

Вариант 1:

Код:

```
switch show_message_ext('Text', 'button 1', 'button 2', 'button 3')
{
  case 1: show_message('button 1'); break;
  case 2: show_message('button 2'); break;
  case 3: show_message('button 3'); break;
}
```

Вариант 2:

Код:

```
var message;
message = show_message_ext('Text', 'button 1', '', 'button 2');
if message == 3
{
  show_message('button 2');
}
```

Вариант 3:

Код:

```
if show_message_ext('Text', 'button 1', '', 'button 2') == 3
{
  show_message('button 2');
}
```

и так далее...

2.1.13. Как выполнить действие, только если объект находится в определённом радиусе?

Ответ: Для того, чтобы проверить, находится ли объект в определённом радиусе - можно использовать функцию point_distance. Проверяя, на каком расстоянии находится объект - мы получим определённый радиус. Пример:

Код:

```
if point_distance(x, y, some_object.x, some_object.y) < 200 // 200, в данном случае, радиус
{
  // Выполняем действия
}
```

2.1.14. Как выполнить действие, только если герой собрал все монеты/ресурсы и т.п.?

Ответ: Для этого нужно проверить кол-во объектов в комнате. Пример:

Код:

```
if instance_number(o_star) == 0
{
  // Выполняем нужные действия
}
```

2.1.15. Я устанавливаю image_speed на единицу, но кадр спрайта меняется мгновенно, а не один раз в секунду! Почему?

Ответ: Присвоив переменной **image_speed** единицу, вы устанавливает смену кадра спрайта каждый шаг. То есть, если у вас скорость комнаты - 30 шагов, то за секунду у спрайта сменится 30 кадров. Если вам нужно, чтобы за секунду менялся один кадр, пишите:

Код:

```
image_speed = 1/room_speed;
Тогда каждую секунду кадр будет изменяться на 1/30 (если у вас скорость комнаты - 30 шагов), и за секунду кадр всего изменится на 30/30, то есть, на 1.
```

2.1.16. Я устанавливаю таймер на единицу, но действие таймера происходит мгновенно, а не через одну секунду! Почему?

Ответ: Таймеры измеряется в шагах, и установив таймеру время в единицу - он выполнится через один шаг, то есть, через 1/**room_speed** секунды. Для того, чтобы таймер выполнился через одну секунду, нужно писать так:

Код:

```
alarm[0] = room_speed;;
```

Таким образом, событие таймера выполнится через room_speed шагов, а room_speed шагов составляют одну секунду.

2.2.1.Как сделать, чтобы при падении игрока сверху на монстра - монстр умирал, а просто при столкновением его с монстром - умирал игрок?

Ответ: При столкновении с монстром нужно проверять позицию игрока. Если игрок выше монстра - убиваем монстра, иначе - наносим урон игроку. Пример, при столкновении игрока с монстром:

Код:

```
if vspeed > 0 and bbox_bottom > other.bbox_top then
{
  with other instance_destroy();
}
else
{
  health -= 4;
}
```

2.2.2. Как сделать фон, двигающийся относительно позиции игрока?

Ответ: Пример, **[end step event]**:

Код:

```
background_x[0] = view_xview[0] *
0.75;
```

2.2.3. Как заставить следовать один объект за другим?

Ответ: Если вы хотите, чтобы объект не останавливался при столкновении с другими объектами - пропишите ему в **step event** такой код:

Код:

```
speed = скорость;
direction = point_direction (x, y, объект.x, объект.y);
```

Если вы хотите, чтобы объект останавливался при контакте с любым, либо только с твёрдым объектом - используйте такой код:

Код:

```
mp_linear_step(объект.x, объект.y, скорость, checkall);
```

Если вместо **checkall** вы впишите **true**, то экземпляр объекта остановится при контакте с экземпляром любого объекта. Если **false**, то он остановиться только при контакте только с твёрдыми экземплярами объекта.

Если же вы хотите, чтобы объект огибал препятствия на пути к цели - используйте функцию **mp_potential_step** аналогично предыдущему способу.

2.2.4. Как сделать разгон для машины, как сделать чтобы игрок постепенно разгонялся и тормозил?

Ответ: На самом деле, на разгон и торможение машины влияет масса показателей. Для общего случая - нужно задать скорость разгона, силу торможения, силу трения и максимальную скорость движения (это касается любого абстрактного игрока, как машини так и реального героя).
Простая схема механизма разгона / торможения для машины будет выглядеть примерно так:

Код:

```
if keyboard_check(vk_up)
{
  скорость += скорость_разгона; // Увеличиваем скорость машины.
}
else if keyboard_check(vk_down)
{
  скорость -= сила_торможения; // Уменьшаем её скорость.
}
else
{
  скорость -= сила_трения;
}

скорость = median(0, скорость, максимальная_скорость); // Ограничиваем скорость, чтобы она не могла быть отрицательной и не превышала максимальный порог.
```

Управление игроком в плафтормере будет выглядеть несколько иначе, хотя и сохраняет основную схему:

Код:

```
// В платформере скорость может принимать отрицательное значение - движение влево (соответственно положительное - движение вправо).
if keyboard_check(vk_left)
  spd = median(-speed_max, spd - 0.5 , speed_max);
else if keyboard_check(vk_right)
  spd = median(-speed_max, spd + 0.5 , speed_max);
else
  spd = max(abs(spd) - 0.2, 0) * sign(spd); // Короткая запись довольно большого куска кода. Если вы не можете понять его значение - вы можете увидеть расшифровку чуть ниже.

if place_free(x + spd, y) // Если ничто не преграждает путь игроку
{
  x += spd; // Двигаем игрока
}
else
{
  var dir;
  if spd > 0 // Самый простой код, позволяющий определить направление движения.
```

```
  dir = 0;
else
  dir = 180;
   move_contact_solid(dir, speed_max); // Пододвигаем игрока вплотную к стене.
   spd = 0; // Обнуляем скорость.
}
```

speed_max в этом коде - максимальная скорость игрока по горизонтали.
spd - текущая скорость игрока.

Эта строка:

Код:

```
spd = max(abs(spd) - 0.2, 0) * sign(spd);
```

Скрывает в себе весь этот код:

Код:

```
if spd > 0
{
   spd -= 0.2;
if spd < 0
  spd = 0;
}
else
{
   spd += 0.2;
if spd > 0
  spd = 0;
}
```

То есть, она уменьшает скорость передвижения героя, сохраняя направление (знак числа).

2.2.5. При повороте мой герой застревает в стене. Я использую "image_angle = point_direction(x, y, mouse_x, mouse_y);", в чём причина?

Ответ: Переменная **image_angle** поворачивает не только картинку, но и маску объекта. Чтобы маска не поворачивалась - нужно использовать свою переменную, к примеру, angle. Пример:

[create event]:

Код:

```
angle = 0;
```

[step event]:

Код:

```
angle = point_direction(x, y, mouse_x, mouse_y);
```

[draw event]:

Код:

```
draw_sprite_ext(sprite_index, image_index, x, y, image_xscale, image_yscale, angle, image_blend,
image_alpha);
```

2.2.6. Почему быстрые пули могут пролетать через стенку?

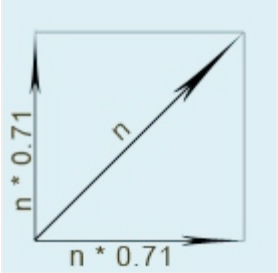
Ответ: Для начала, нужно понять, почему это происходит. Дело в том, что за один шаг пуля мгновенно перемещается на некоторое расстояние, хотя столкновение проверяется только в новой позиции - при большой скорости пуля может запросто "перескочить" стенку.
Как вариант, можно использовать такой код в **step** пули:

Код:

```
var i;
for (i = sprite_width; i <= speed; i += sprite_width)
{
  if place_meeting(x + lengthdir_x(i,direction), y + lengthdir_y(i,direction), o_wall) // Если в этой позиции есть
стена, то...
  {
    x += lengthdir_x(i, direction); // Двигаемся к позиции столкновения
    y += lengthdir_y(i, direction);
    event_perform(ev_collision, o_wall); // Вызываем событие столкновения со стеной
  }
}
```

2.2.7. Как сделать, чтобы игрок в tds двигался по диагонали с той же скоростью, что и по прямой?

Ответ: Проблема состоит в том, что диагональ квадрата больше стороны квадрата. При чём тут квадрат? А вот в при чём: когда сдвигаем героя просто вверх, просто влево, вправо или вниз - он сдвигается на n пикселей, когда же мы сдвигаем его по диагонали - наш герой в действительности сдвигается на размер диагонали квадрата со стороной в n пикселей (к примеру, если мы сдвигаем героя на 100 пикселей влево и 100 пикселей вправо - герой в действительности сдвигается на ~141 пиксель). Соотношение диагонали квадрата к его стороне примерно равно 0.707. Чтобы получить нормальный сдвиг - нужно поделить скорость на корень из двух (~1.41) или умножить на 0.707 (1/sqrt(2)).



Скачать пример: tds move.gmk

2.2.8. Как сделать плавный поворот турели?

Ответ: Для этого нужно добавить два скрипта. Первый, **angle_difference** - вычисляет разницу между двумя направлениями:

Код:

```
/*
**  usage:
**      diff = angle_difference(angle1,angle2);
**
**  given:
**      angle1    first direction in degrees, real
**      angle2    second direction in degrees, real
**
**  returns:
**      difference of the given angles in degrees, -180 to 180
**
**  GMLscripts.com
*/
{
  return ((((argument0 - argument1) mod 360) + 540) mod 360) - 180;
}
```

Второй, **set_angle** (здесь название может быть любым) - вычисляет новое направление:

Код:

```
/*
Скрипт плавно изменяет направление башни до требуемого
Возвращает полученное направление
argument0 - текущее направление
argument1 - требуемое направление
argument2 - скорость изменения направления
Автор скрипта DreamRunner 09.09.2006
*/

argument0 = argument0 mod 360; // Ограничиваем направление - от -360 до 360 (не включительно).
if argument0 <0 argument0 += 360; // Если направление меньше 360 - преобразовываем значение в положительное
if abs(angle_difference(argument0,argument1)) < argument2 // Если разница между направлениями меньше скорости поворота, то...
  return argument1; // Возвращаем требуемое направление
return argument0 + (sign(sin(degtorad(argument1-argument0))) * argument2);
/*
sign(sin(degtorad(argument1-argument0))) - от 0 до 180 получаем положительно значение, от 180 до 360 - отрицательное.
т.е. определяем, в какую сторону нужно повернуть объект. Если разница между направлениями меньше 180 - значит по часовой стрелке, больше - против часовой.
*/
```

Пример использования, **[step event]**:

Код:

```
image_angle = set_angle(image_angle, point_direction(x, y, mouse_x, mouse_y), 3);
```

2.2.9. Как сделать движущиеся платформы в платформере?

Ответ: На самом деле, в этом нет ничего сложного, нужно только хорошенько подумать. Если вы используете встроенные механизмы движения, вроде **vspeed** и **gravity** - позиция игрока измениться после **step event** и до **end step event**. Значит, платформа, как и игрок сдвинется после step'а игрока. Итак, в **step event** игрока мы проверяем, нет ли платформы под игроком. Если под игроком есть подвижная платформа - мы записываем её **id** в переменную, чтобы двигать игрока за ней. В **end step** игрока мы сдвигаем игрока относительно платформы, на которой он стоит. Далее мы проверяем, если игрок всё же столкнулся с платформой (к примеру, когда горизонтальная платформа наехала на игрока) - мы возвращаем платформу на предыдущую позицию и меняем её направление движения - буквально "выталкиваем" из игрока. В коде это выглядит примерно так:

[create event] игрока:

Код:

```
platform = noone; // Изначально игрок не стоит на подвижной платформе
```

[step event] игрока:

Код:

```
//...
platform = instance_place(x, y + 1, o_moving_platform); // Определяем id платформы под игроком, если под игроком нет платформы - функция возвратит значение noone, -4. id всегда начинается с 100001, что значит - можно использовать проверку "if platform", что аналогично "if platform >= 0.5". Если для вас эта проверка в новинку, можете писать "if platform != noone"
```

[end step event] игрока:

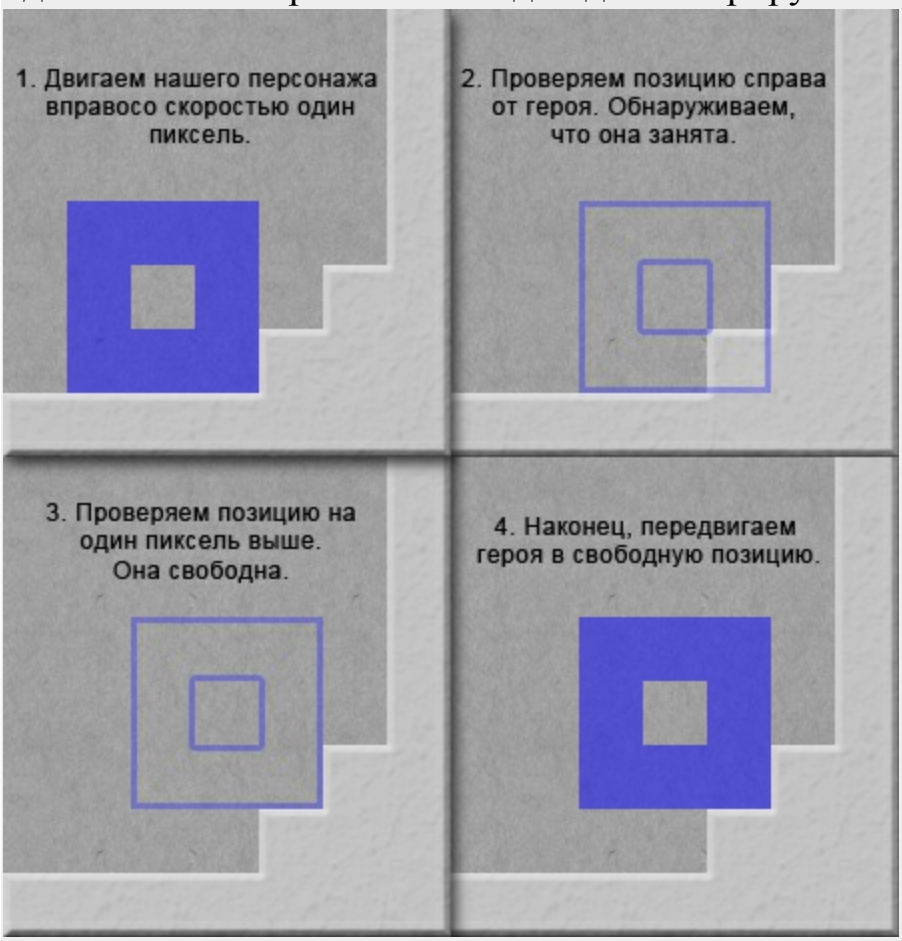
Код:

```
//...
if platform // Если игрок стоит на платформе, то...
{
  x += platform.hspeed; // Двигаем игрока относительно платформы
  y += platform.vspeed;
}

platform = instance_place(x, y, o_moving_platform);
if platform // Если на игрока наехала платформа, то...
{
  with platform
  {
    x = xprevious; // Возвращаем платформу на предыдущую позицию
    y = yprevious;
    hspeed *= -1; // Инвертируем скорость, если платформа двигалась влево - она будет двигаться вправо и т.п.
    vspeed *= -1;
  }
}
```

P.S: если у вас возникают проблемы со столкновениями, залипания игрока в платформах и т.п. - советую вам отказаться от использования **vspeed** и **hspeed**, так как встроенный механизм движения автоматически останавливает объект при столкновении с solid объектом.

2.2.10. Как сделать движение по неровной поверхности в платформере?

Ответ: Здесь тоже нет ничего сложного. Проблема в том, что иногда игроку может препятствовать всего один пиксель, и чтобы на него взобраться - приходится прыгать. Потому, нужно создать некоторую имитацию "ног человека" - за один шаг игрок может взобраться на несколько пикселей, без прыжка. Итак, сначала задаётся максимальная высота, на которую может взобраться игрок - обычно 4-8 пикселей, во время движения в цикле проверяется сначала позиция прямо возле игрока, потом на один пиксель выше, на два пикселя выше, и так до максимальной высоты, на которую может взобраться игрок за один шаг. Эта картинка наглядно демонстрирует метод:



В коде это будет выглядеть примерно так:
[step event]:

Код:

```
var i;
if keyboard_check(vk_left) // Если игрок нажал стрелку влево
{
  for (i = 0; i < vdis_max; i += 1) // В цикле проверяем все позиции: с высотой как у игрока, на 1 пиксель выше, на 2 и т.д.
```

```
  {
    if place_free(x - spd, y - i) // Если данная позиция свободна, то...
    {
      x -= spd; // Двигаем игрока
      y -= i;
      break; // Завершаем цикл
    }
  }
}

//...
```

Где **vdis_max** - максимальная высота, на которую может взобраться игрок за один шаг, **spd** - скорость передвижения игрока.
В случае, если нужно реализовать ещё и быстрый спуск по ступенькам вниз - код будет выглядеть примерно так:

Код:

```
var i;
if keyboard_check(vk_left) // Если игрок нажал стрелку влево
{
  for (i = 0; i < vdis_max; i += 1) // В цикле проверяем все позиции: с высотой как у игрока, на 1 пиксель выше, на 2 и т.д.
  if place_free(x - spd, y - i) // Если данная позиция свободна
  {
    x -= spd; // Двигаем игрока
    y -= i;

    if place_free(x, y + 1) // Если игрок не стоит на платформе,
    if !place_free(x, y + vdis_max2 + 1) // Но она под ним есть!
    for (i = 0; i < vdis_max2 + 1; i += 1) // Проверяем все позиции немного ниже его
    {
      if !place_free(x, y + i) // Если в данная позиция занята твёрдым объектом
      {
        y += i - 1; // Двигаем игрока вплотную к этому объекту
        break; // Завершаем цикл
      }
    }
    break; // Завершаем цикл
  }
}
```

Собственно, эти циклы - самописные аналоги функций **move_outside_solid** и **move_contact_solid**. При желании их можно заменить встроенными функциями.

2.2.11. Как сделать, чтобы игрок запрыгивал на платформы снизу?

Ответ: Идея проста: выполнять событие столкновения с платформой только если игрок находится выше платформы. Переменная **bbox_bottom** - нижняя граница прямоугольника, ограничивающего спрайт объекта. **bbox_top**, соответственно, верхняя. Можно было бы просто сравнить **bbox_bottom** игрока и **bbox_top** платформы - но часто бывает так, что событие столкновения вызывается уже тогда, когда игрок

непосредственно пересекается с платформой, и проверка может не сработать. Потому, поверку в событии столкновения игрока с платформой можно записать так:

Код:

```
if bbox_bottom-vspeed <= other.bbox_top // Если в предыдущей позиции игрок был выше платформы, а теперь столкнулся с ней, то...
{
  // Выполняем события столкновения... (можно вернуть игрока на предыдущую позицию, и с помощью move_contact_solid сдвинуть его вплотную к платформе)
}
```

Так же обратите внимание, что объекты, которые движутся с помощью стандартных механизмов движения GM - автоматически останавливаются при контакте с solid объектами.

2.2.12. Как сделать, чтобы игрок скользил по неровным стенам в tds игре?

Ответ: Используется тот же принцип, что и с движением по неровной поверхности в платформере, только реализуется немного иначе. Если позиция перед игроком занята - пытаемся изменить направление игрока так, чтобы позиция была свободна - на 10 градусов влево/вправо, на 20 градусов влево/вправо и т.д.
Пример реализации:

Код:

```
var spd, dir, angle, angle_step, lx, ly, i;
spd = 8; // Скорость движения игрока
dir = -1;
angle = 90; // Максимальный поворот игрока
angle_step = 5; // Величина, на которую увеличивается направление в шаге цикла

if keyboard_check(ord('W')) dir = 90; // В зависимости от того, какую клавишу нажал пользователь - определяем направление игрока
if keyboard_check(ord('A')) dir = 180;
if keyboard_check(ord('S')) dir = 270;
if keyboard_check(ord('D')) dir = 0;

if dir != -1 // Если игрока нужно подвинуть, то...
{
  for (i = 0; i < angle; i += angle_step) // Цикл отклонения. В начале - проверяем позицию прямо перед игроком, затем +-angle_step, +-angle_step*2 и т.д.
  {
    lx = lengthdir_x(spd, dir + i); // Вычисляем смещение игрока
    ly = lengthdir_y(spd, dir + i);
    if place_free(x + lx, y + ly) // Если эта позиция свободна, то...
    {
      x += lx; // Двигаем игрока
      y += ly;
      break; // Завершаем цикл
    }

    lx = lengthdir_x(spd, dir - i); // Проверяем позицию левее; действия и проверки аналогичны
    ly = lengthdir_y(spd, dir - i);
    if place_free(x + lx, y + ly)
    {
```

```
            x += lx;
            y += ly;
            break;
        }
    }
}
```

Для ускорения работы алгоритма можно увеличить значение **angle_step** и/или уменьшить значение **angle**. Для увеличения точности нужно уменьшать значение **angle_step**

2.2.13. Как прикрепить башню танка к самому танку?

Ответ: Во-первых, позицию башни танка нужно изменять исключительно после изменения позиции танка, чтобы она не отставала. Прикрепить башню можно двумя способами, либо написать примерно такой код в **step event** башни:

Код:

```
x = o_panzer.x; // Изменяем позицию башни
y = o_panzer.y;
```

Либо в **draw event** сразу рисовать спрайт башни в нужной позиции:

Код:

```
draw_sprite_ext(sprite_index, image_index, o_panzer.x, o_panzer.y, image_xscale, image_yscale, image_angle, image_blend, image_alpha);
```

2.2.14. Как сделать, чтобы интерфейс не отставал от вида при движении?

Ответ: Позицию элементов интерфейса нужно менять после изменения позиции объекта, за которым следит вид - в **end step event** либо в **draw event**.

2.2.15. Как определить позицию столкновения? Как определить, в какую сторону ударил снаряд?

Ответ: Определить позицию столкновения часто бывает довольно сложно, если снаряд и противник имеют сложную форму. Вычислить позицию столкновения можно двумя способами: относительно пули, и относительно объекта, с котором столкнулась пуля. Если пуля круглая или всё время повёрнута в сторону полёта - проще всего. Для начала нам нужно определить расстояние от центра спрайта до точки столкновения при повороте 0 градусов (для круглой пули поворот не учитывается). Если пуля использует **bbox** для проверки столкновений - ограничивающий прямоугольник, формула может выглядеть так:

Код:

```
len = sprite_get_bbox_right(sprite_index) - sprite_get_xoffset(sprite_index)
```

Событие столкновения часто вызывается уже тогда, когда пуля непосредственно над объектом. Потому, перед поиском точки столкновения нужно выполнить такой код:

Код:

```
x = xprevious; // Возвращаем пулю на предыдущую позицию
y = yprevious;
move_contact_solid(direction, speed); // Пододвигаем её вплотную к объекту (объект должен быть твёрдым)
```

Итак, если направление пули - **direction**, код будет выглядеть так:

Код:

```
var len, xx, yy;
x = xprevious;
y = yprevious;
move_contact_solid(direction, speed);

len = sprite_get_bbox_right(sprite_index) - sprite_get_xoffset(sprite_index);
xx = x + lengthdir_x(len, direction);
yy = y + lengthdir_y(len, direction);
```

Где **xx, yy** - полученная точка столкновения. Если пуля неровная, но объект, с которым она столкнулась повёрнут в ней, либо он круглый - выполняем те же действия, только для объекта-преграды. В случае, когда таким способом точку столкновения обнаружить не удаётся - вам придётся придумывать собственные формулы, для вашего конкретного случая.

2.2.16. Как сделать, чтобы герой в игре жанра "лабиринт" (в игре с видом сверху) мог толкать ящики?

Ответ: Реализация возможности двигать ящики зависит от того, как реализовано управление игроком. Если он двигается за счёт **speed** и **direction**, можно поставить такой код в событие столкновения ящика с игроком:

Код:

```
if place_free(x + lengthdir_x(other.speed, other.direction), y + lengthdir_y(other.speed, other.direction)) // Если ящик можно сдвинуть (позиция свободна)
{
  x += lengthdir_x(other.speed, other.direction); // Двигаем ящик по направлению движения игрока
  y += lengthdir_y(other.speed, other.direction);
}
else // Если ящик нельзя сдвинуть
{
  with other // Возвращаем игрока на предыдущую позицию, чтобы он не застрял в ящике
  {
     x = xprevious;
     y = yprevious;
```

```
    speed = 0;
  }
}
```

Если игрок двигается за счёт **vspeed** и **hspeed**, можно использовать такой код (практически аналогичный предыдущему):

Код:

```
if place_free(x + other.hspeed, y + other.vspeed)
{
  x += other.hspeed;
  y += other.vspeed;
}
else
{
  with other
  {
    x = xprevious;
    y = yprevious;
    vspeed = 0;
    hspeed = 0;
  }
}
```

В случае, если игрок двигается непосредственно через **x** и **y** - можно использовать такой код:

Код:

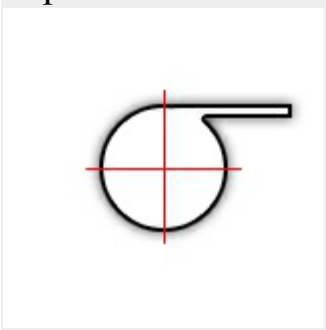
```
var dx, dy; // Объявляем временные переменные
dx = other.x - other.xprevious; // Вычисляем
dy = other.y - other.yprevious;

if place_free(x + dx, y + dy) // Если ящик можно сдвинуть
{
  x += dx; // Двигаем ящик по направлению движения игрока
  y += dy;
}
else // Если ящик нельзя сдвинуть
{
  with other // Возвращаем игрока на предыдущую позицию
  {
    x = xprevious;
    y = yprevious;
  }
}
```

Можно усовершенствовать код, используя функцию **move_contact_solid**, для того, чтобы придвинуть игрока к ящику. Ещё можно вычислять скорость игрока и сдвигать ящик на вдвое меньшее расстояние, чтобы создать эффект тяжести.

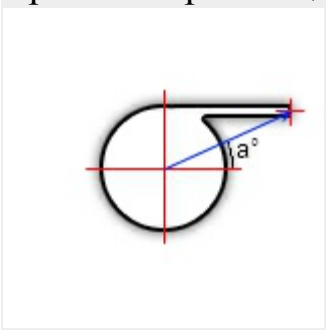
2.2.17. Как сделать, чтобы пуля вылетала из дула?

Ответ: Нужно использовать **lengthdir_x** и **lengthdir_y**. Предположим, у нас есть такой

спрайт:



Красный крест - центр спрайта. Пуля должна создаваться на конце дула:



Для того, чтобы она создавалась в нужном месте нужно:
1) Записать координаты центра спрайта и позиции, в которой нужно создать объект;
2) Вычислить расстояние между этими двумя координатами;
3) Вычислить направление между этими двумя координатами.
Тепрь, пулю можно создавать в этой позиции:

Код:

```
xx = x + lengthdir_x(len, direction + dir);
yy = y + lengthdir_y(len, direction + dir);
```

Где **len** - расстоние между двумя точками, **direction** - текущее направление объекта, **dir** - направление от центра спрайта к точке на конце дула.
Расстояние и направление можно вычислить с помощью функций **point_distance** и **point_direction**.

2.2.18. Как реализовать мгновенные пули?

Ответ: Для реализации мгновенных пуль нужно либо написать свой алгоритм, аналогичный **collision_line**, либо использовать **move_contact_solid**. К примеру, при стрельбе вы могли бы использовать подобный алгоритм:

Код:

```
var i, temp_x, temp_y; // Объявляем временные переменные
for(i = 0; i < shoot_distance; i += 4)
{
  temp_x = x + lengthdir_x(i, direction); // Вычисляем новую позицию
  temp_y = y + lengthdir_y(i, direction);

  obj = collision_circle(temp_x, temp_y, 2, all, 0, 0);
```

```
  if obj != noone // Если в этой позиции находится какой-то объект
  {
    if obj.object_index = o_enemy // Если это враг, то...
    {
      with obj
        hp -= 10; // Уменьшаем его кол-во жизней
    }
    if obj != id // Если это не тот объект, кто стреляет, то...
    {
      break; // Завершаем цикл
    }
  }
}
```

Где **shoot_distance** - максимальная дистанция, которую может достигнуть пуля, direction - направление полёта пули и **o_enemy** - объект врага. Заметьте, что **hp** не изначально определённая переменная, и она использована здесь для примера. Для оптимизации скрипта - можно увеличить кол-во пикселей, на которое смещается позиция проверки каждый шаг (4 по умолчанию), или заменить проверку **collision_circle** чем-то вроде **collision_point**.
Обратите внимание, что проверка производиться для всех объектов, но, если, к примеру, в одной позиции будет одновременно находится объект врага и объект, скажем, решётки на полу - может возникнуть баг, так как пуля может не проверить столкновение с врагом. В таком случае - для всех объектов, с которыми происходит действие пули - можно создать родитель (parent), и заменить **all** в коде на его имя. Другой вариант - делать несколько проверок для каждого объекта.
Стоит учесть, что это далеко не единственный способ реализации мгновенной стрельбы.

2.3.1. Как отобразить объект для конкретного игрока, в случае, если у меня включено два вида?

Ответ: Используйте переменную **view_current**. Пример:

Код:

```
if view_current = 0
{
  draw_text(view_xview[view_current], view_yview[view_current], 'Игрок 1');
}

if view_current = 1
{
  draw_text(view_xview[view_current], view_yview[view_current], 'Игрок 2');
}
```

2.3.2. Я отрисовываю на экране, к примеру, текст, но когда я двигаюсь - он улетает за границы экрана. Как сделать так, чтобы текст оставался в исходной позиции?

Ответ: На самом деле, позиция текста не меняется, однако координаты вида, напротив, изменяются. Вид - область, которую нужно отрисовать на экране. Чтобы отрисовать что-либо на виде - к координатам x и y нужно прибавлять **view_xview** и **view_yview** соответственно. Например:

Код:

```
draw_text(view_xview + 16, view_yview + 32, "Text");
```

Чтобы отрисовывать текст, например, над главным героем - можно написать так:

Код:

```
draw_text(o_player.x + 16, o_player.y - 32, "Text");
```

2.3.3.Я пишу в событие draw event в объекте игрока отображение жизней (например). Но в игре спрайт игрока не отображается. В чём причина?

Ответ: Если событие draw event пусто - GM автоматически отрисовывает спрайт объекта, в случае, если он находится в пределах вида. Если же вы задали в этом событии какие-то действия - GM не может знать, находится ли ваша графика в пределах вида, или вне его. Потому, эти действия выполняются в любом случае, а спрайт объекта не отображается. Чтобы нарисовать спрайт вручную - нужно дописать такой код в начало события **draw event**:

Код:

```
draw_sprite_ext(sprite_index, image_index, x, y, image_xscale, image_yscale, image_angle, image_blend,
image_alpha);
```

Эта функция использует встроенные переменные для изменения масштаба, прозрачности, поворота и тд. спрайта, но если вы не используете эти функции - вы можете писать просто

Код:

```
draw_sprite(sprite_index, image_index, x, y);
```

2.3.4. Как изменить глубину объекта в draw event?

Ответ: Глубину объекта прямо в событии **draw** изменить нельзя. Можно использовать несколько объектов с разной глубиной или просто переставить действия в событии рисования (если вы используете 3D - почитайте про d3d_set_depth(depth)).

2.3.5: Вопрос 1: Как сделать, чтобы когда герой был ниже дерева - он был над ним, а когда выше - за ним? Вопрос 2: Какую нужно использовать формулу, чтобы вычислить глубину объекта в изометрии или в играх с видом как в классических jrpg?

Ответ: Для игр с видом как в классических jrpg обычно используется формула **depth=-y** при условии, что центр спрайта находится в точке столкновения объекта с землёй, а не в (0,0). Эту код нужно поставить в **create event** статических объектов, и step event динамических. Чем ниже объект, тем меньше его глубина, тем выше он рисуется.
Для изометрии можно использовать формулу **depth = -(y - sprite_yoffset + (sprite_height + H) / 2)**, где H - высота объекта над уровнем земли. Если у одного и того же спрайта высота точек спрайта над уровнем земли разная - нужно использовать буфер глубины, то есть - 3D технологии.

2.3.6. Как сделать ссылку / Текст, при клике на который выполнялось бы определённое действие?

Ответ: Можно использовать следующий скрипт:

Код:

```
/*************************************************
 Скрипт рисует ссылку.
 argument0 - x;
 argument1 - y;
 argument2 - текст;
 Возвращает 1, если игрок кликнул на текст, иначе - 0.
*************************************************/
var xx, yy, str, width, height; // Задаём временные переменные
xx = argument0;
yy = argument1;
str = argument2;
width = string_width(str);
height = string_height(str);
```

```
if mouse_x > xx and mouse_y > yy and mouse_x < xx + width and mouse_y < yy + height // Если пользователь навёл на мышку, то...
{
  draw_set_color(c_red); // Устанавливаем цвет, с которым рисуется выделенная ссылка
  draw_text(xx, yy, str); // Рисуем текст
  draw_line(xx, yy + height, xx + width, yy + height); // Рисуем подчёркивание

  if mouse_check_button_released(mb_left) // Если пользователь кликнул на ссылку, то...
  {
     return 1; // Возвращаем 1, и завершаем работу скрипта
  }
}
else
{
  draw_set_color(c_blue); // Рисуем ссылку обычным цветом
  draw_text(xx, yy, str);
}

return 0; // Возвращаем 0 - пользователь не кликал на ссылку
```

Пример использования, **[draw event]**:

Код:

```
if draw_text_button(0, 0, 'button')
{
  execute_shell("http://google.com/", 0);
}
```

Для открытия web страницы в игровом окне можно использовать функцию **splash_show_web**.

2.3.7. Можно ли для хелсбара использовать спрайт, а не цвет?

Ответ: Конечно!

Код:

```
draw_sprite(s_healthbar, 0, view_xview + 8, view_yview + 8)  //Рисуем фон
draw_sprite_part(s_healthbar, 1, 0, 0, sprite_get_width(s_healthbar) * (Health / Health_max),
sprite_get_height(s_healthbar), view_xview + 8, view_yview + 8) //Рисуем жизни на фоне
```

s_healthbar - спрайт хелсбара, первый кадр должен быть фоном хелсбара, а второй кадр - картинка полностью заполненного жизнями хелбара.
Health - текущее кол-во жизней, **Health_max** - максимальное кол-во жизней.
view_xview + 8, **view_yview** + 8 - позиция хелсбара

2.3.8. Как сделать, чтобы при наведении на объект высвечивалась информация о нём?

Ответ: В **draw event** объекта, при наведении на который должна высвечиваться

информация следует добавить такой код:

Код:

```
if position_meeting(mouse_x, mouse_y, id) // Если в позиции мышки находится данный экземпляр объекта, то...
{
  // Рисуем нужную информацию.
}
```

Так же не стоит забывать, что если добавить какой-то код в событие **draw** - GM перестанет автоматически рисовать спрайт объекта, так что его отрисовку нужно добавить вручную (с помощью функции **draw_sprite**, к примеру).

2.3.9. Как повернуть сурфейс относительно его центра?

Ответ: Нужно использовать 3D-преобразования, которые отлично работают и в 2D. Пример:

Код:

```
d3d_transform_add_translation(-surface_get_width(surface)/2, -surface_get_height(surface)/2, 0); // Устанавливаем сурфейс в центр
d3d_transform_add_rotation_z(image_angle); // Поворачиваем его на величину image_angle
d3d_transform_add_translation(x, y, 0); // Сдвигаем в нужную позицию, там, где будет рисоваться сурфейс
draw_surface(surface, 0, 0); // Рисуем сурфейс
d3d_transform_set_identity(); // Отключаем все преобразования, установленные пользователем
```

2.3.10. Как полностью перекрасить спрайт в другой цвет?

Ответ: Для того, чтобы перекрасить спрайт в чёрный цвет - можете использовать **image_blend**, тот же параметр **color** в функциях для отображения спрайтов. Кстати, это довольно полезная функция, к примеру - с её помощью можно легко создать тень, пример:

Код:

```
draw_sprite_ext(sprite_index, 0, x + 2, y + 2, 1, 1, 0, c_black, 0.3); // Рисуем тень с непрозрачностью 0.3
draw_sprite(sprite_index, 0, x, y); // Рисуем сам спрайт
```

Чтобы полность перекрасить спрайт в цвет, отличный от чёрного - используйте функцию для создания тумана в 3D играх. Пример:

Код:

```
d3d_set_fog(true, c_white, 0, 0);
draw_sprite(sprite_index, 0, x, y);
d3d_set_fog(false, c_white, 0, 0);
```

Ну, а если вы хотите создать полноценный эффект "Fade" - используйте dll для работы с пиксельными шейдерами в GM, где это реализуется очень легко:

http://gmc.yoyogames.com/index.php?showtopic=492876

2.3.11. Как убрать курсор с помощью кода? Как изменить курсор? Как сделать свою картинку курсора?

Ответ: Изменить стандартное изображение можно с помощью функции **window_set_cursor(curs)**, в которой можно использовать следующие константы:

Код:

```
cr_default
cr_none
cr_arrow
cr_cross
cr_beam
cr_size_nesw
cr_size_ns
cr_size_nwse
cr_size_we
cr_uparrow
cr_hourglass
cr_drag
cr_nodrop
cr_hsplit
cr_vsplit
cr_multidrag
cr_sqlwait
cr_no
cr_appstart
cr_help
cr_handpoint
cr_size_all
```

Чтобы убрать курсор - следует выбрать константу **cr_none**.
Чтобы показывалась картинка игрового курсора, можно использовать переменную **cursor_sprite**.
Обратите внимание, если скорость комнаты меньше 60 - ваш собственный курсор будет немного "тормозить", потому что частота обновления экрана значительно больше 30.

2.4.1. Как добавить в игру ману? Как отрисовать кол-во маны на экране?

Ответ: Для жизней можно использовать стандартную переменную **health**. А вот для маны потребуется завести свою. В событии **create event** поставьте серый квадратик с именем **"VAR" (control -> Variables -> Set Variable)**. В первом поле (**variable**) запишите название своей переменной на английском языке, без пробелов и начиная с буквы, например, **mana**. Если вы хотите изменить значение переменной (в данном случае - маны, к примеру, после использования эликсира) - поставьте тот же квадратик, и введите в него имя переменной и требуемое значение. Вариант кодом: «**mana = 0**;». Чтобы отобразить значение переменной на экране (в данном случае - количество маны), в событие **draw event** поставьте светло-жёлтый квадратик с именем **"VAR" (control -> Variables -> Draw Variable)**, в нём введите имя вашей переменной и желаемые координаты. Возможный вариант кодом:

Код:

```
draw_text(/*ваши координаты*/, /*ваши координаты*/, "мана: " + string(mana));
```

2.4.2. Как мне изменить вид стандартных серых окон GM?

Ответ: Используйте функции **message_***:

Код:

```
message_background(back)
message_alpha(alpha)
message_button(spr)
message_text_font(name,size,color,style)
message_button_font(name,size,color,style)
message_input_font(name,size,color,style)
message_mouse_color(col)
message_input_color(col)
message_caption(show,str)
message_position(x,y)
message_size(w,h)
```

P.S: за описанием - в справку.

2.4.3. Хелсбар не показывает больше 100 жизней, хотя health у меня равно 200! Как мне установить максимальное значение health?

Ответ: В функции **draw_healthbar** в аргументе **amount** нужно использовать такую формулу:
(health / health_max) * 100
Где **health** - переменная жизней, **health_max** - максимальное кол-во жизней.

2.4.4. Как определить наивысший объект под мышкой?

Ответ: Скрипт опеределения наивысшего объекта в конкретной позиции выглядит так:

Код:

```
/*************************************************
 Скрипт определяет наивысший экземпляр объекта в указанной позиции.
 Возвращает id экземпляра объекта, если он существует, в противном случае - noone.
 argument0 - x;
 argument1 - y;
 argument2 - объект;
*************************************************/
var high_object;
high_object = noone;
with argument2 // Обращаемся ко всем экземплярам объекта
{
  if position_meeting(argument0, argument1, id) // Если объект находится в нужной позиции, то...
  {
    if other.high_object = noone // Если мы ещё не выбрали ни один объект, то...
    {
      other.high_object = id; // Запоминаем текущий объект.
    }
    else
    {
      if depth < other.high_object.depth  // Если текущий объект выше чем выбранный, то...
      {
        other.high_object = id; // Запоминаем объект.
      }
    }
  }
}

return high_object; // Возвращаем id найденного объекта.
```

В этом скрипте идёт обращение ко всем экземплярам искомого объекта. Вначале каждый из них проверяется на столкновение с указанной позицией, то есть, проверяется, находится ли он в ней. Если да, тогда его глубина сравнивается с глубиной записанного объекта (тут и далее подразумевается экземпляр объекта). Если его глубина меньше - этот объект заменяет ранее записанный. Конечно, если ни один объект ещё не был записан - глубина не проверяется, а объект сразу записавается.

Скачать пример.

2.4.5. Почему при повороте спрайта с использованием image_angle его изображение искажается и выглядит некрасиво?

Ответ: Включите в настройках игры интерполяцию **(global game setting -> graphics -> interpolate color between pixel)**.

2.4.6. Как мне нарисовать русский текст? Почему я его не вижу?

Ответ: В Game Maker по умолчанию встроен шрифт **Arial 12**, с ограниченным набором символов: начиная от 32 до 127. Русский алфавит в этот диапазон не входит, поэтому, чтобы рисовать русский текст, вам понадобится добавить свой шрифт, в котором нужно выставить максимальный диапазон символов, для поддержки кириллицы. Итак, выберите пункт "**Create Font**" из меню "**Resources**", или просто нажмите соответствующую кнопку на панели инструментов. Затем дайте имя своему шрифту в игре (я лично предпочитаю что-то вроде **font_menu_button** или **font_arial10b**), выберите сам шрифт, установите нужный размер и нажмите на кнопку "**All**" (Все), которая устанавливает максимальный диапазон символов. Наконец, вам нужно установить для использования этот шрифт в игре, для этого перед рисованием текста вызовите функцию **draw_set_font**:

Код:

```
draw_set_font(имя_вашего_шрифта_в_игре);
draw_text(...);
```

Если вы всё равно не увидили русские буквы - значит выбранный вами шрифт не поддерживает кириллицу.

2.4.7. Скорость смены кадров у меня 0.3. Я хочу, чтобы на втором кадре появлялся взрыв. Для этого я пишу: "if image_index = 2 then instance_create(x, y, o_explosion);" Но взрыв не появляется! В чём причина?

Ответ:Перед тем, как я покажу вам путь решения - я хочу уведомить вас о ошибке, сделанной в официальном справочном руководстве (а точнее - в GM). Во главе "Спрайты и изображения", подраздела "Игровая графика" раздела "Game Maker Language (GML)" мы можем увидеть такую строчку:

> **image_index** Содержит номер кадра спрайта текущего экземпляра объекта, с которого следует начать проигрывание анимации. Эта переменная указывает к настоящему времени нарисованный кадр (нумерация начинается с 0). Вы можете изменить текущее изображение изменением этой переменной. Программа продолжит повторение, начиная с новым индексом. (Значение может иметь дробную часть. В этом случае, оно всегда будет округлено в меньшую сторону, чтобы получить кадр, который рисуется.)

Однако, тут допущена важная ошибка, так как округление производится «правильное», а не в меньшую сторону. Т.е. если, скажем, **image_index** равно **1.4**, то после округления мы получим **1**, а если оно равно **1.5** - получим **2**. Это важно учитывать, так как мы увидим второй кадр на экране, когда округлённое значение **image_index** фактически будет равно 1 (нумерация начинается с **0**, так что второй кадр - **1**). А потому, следует написать так:

Код:

```
if round(image_index) = 1 then instance_create(x, y, o_explosion);
```

P.S: если вас смущают русские цитаты - загляните в английскую справку, это не ошибка переводчиков.

2.4.8. Вопрос 1: Как запросить у игрока код, к примеру, для открытия двери? Вопрос 2: Как сделать чит-коды?

Ответ: Обе задачи решаются очень просто, с помощью функции **get_string**. Запросить код можно следующим образом:

Код:

```
var text;
text = get_string('Введите код для открытия двери:', '****'); // Запрашиваем у игрока код.
if text = '1234' // Сравниваем его с правильным кодом.
{
  // Действия, которые выполняются, если игрок ввёл правильный код.
}
else
{
  show_message('Код неверен.'); // Выводим сообщение об ошибке.
}
```

Чит-коды можно организовать следующим образом:

Код:

```
var text;
text = get_string('Введите чит-код:', ''); // Запрашиваем у игрока чит-код.
switch (text) // Сравнивем введённый игроком текст со всеми чит-кодами. Не забудьте break в конце каждой проверки!
{
  case 'lives': lives = 100; break;
  case 'gold': score += 1000000; break;
  default: show_message('Такого чит-кода не существует!');
}
```

2.4.9. Как изменить размер текущей комнаты?

Ответ: Размер текущей комнаты напрямую изменить нельзя. Как вариант, можно перейти в другую комнату, изменить размер нужной комнаты, а затем вернуться в изначальную. Чтобы не перезапускать комнату - можно использовать свои переменные для хранения и изменения её размера. К примеру, в каком-нибудь контроллере объявляем глобальные переменные **level_width** и **level_height**. **room_width** и **room_height** везде заменяем на **level_width** и **level_height** соответственно. После этого нужно написать свой алгоритм движения вида - в событии **begin step** сначала двигаем объект, за которым следит вид (к примеру, героя), а затем сдвигаем вид по такому алгоритму:

Код:

```
var xx, yy;
xx = x; // Устанавливаем позицию, к которой должен двигаться вид
yy = y;
```

```
if xx < view_xview[0] + view_hborder[0]
   view_xview[0] -= min(view_hspeed[0], view_xview[0] + view_hborder[0] - xx);
if yy < view_yview[0] + view_vborder[0]
   view_yview[0] -= min(view_vspeed[0], view_yview[0] + view_vborder[0] - yy);

if xx > view_xview[0] + view_wview[0] - view_hborder[0]
   view_xview[0] += min(view_hspeed[0], xx - (view_xview[0] + view_wview[0] - view_hborder[0]));
if yy > view_yview[0] + view_hview[0] - view_vborder[0]
   view_yview[0] += min(view_vspeed[0], yy - (view_yview[0] + view_hview[0] - view_vborder[0]));

view_xview[0] = median(0, view_xview[0], level_width - view_wview[0]); // Ограничиваем позицию вида
view_yview[0] = median(0, view_yview[0], level_height - view_hview[0]);
```

Где **xx** и **yy** - координаты точки, за которой следит вид. Все настройки вида в комнате можно оставить без изменений, только нужно обязательно убрать объект, за которым следует вид - установаить **<no object>** (опция находится в закладке **view**, **object following**). Теперь, можно свободно изменять переменные **level_width** и **level_height**, не перезапуская комнату.

2.4.10. Как определить время компьютера? Как отобразить текущее время на экране?

Ответ: Все функции можно посмотреть в справке: GML -> Игровой процесс -> Синхронизация:

> **current_time*** Количество миллисекунд, которые прошли с тех пор, как система была запущена.
> **current_year*** Текущий год.
> **current_month*** Текущий месяц.
> **current_day*** Текущий день.
> **current_weekday*** Текущий день недели (1=воскресенье, ..., 7=суббота).
> **current_hour*** Текущий час.
> **current_minute*** Текущая минута.
> **current_second*** Текущая секунда.

Отобразить дату и время компьютера можно следующим образом:

Код:

```
draw_text(x, y, string(current_hour) + ':' + string(current_minute) + ':' + string(current_second) + ' / ' +
string(current_day) + '.' + string(current_month) + '.' + string(current_year));
```

2.4.11. Как защитить счёт в игре от артмани?

Ответ:

Метод 1:

Можно хранить в памяти не то значение которое выводится на экран - тогда пользователь не сможет узнать, по какому именно значению проводить поиск.

Пример реализации:

[create event]:

Код:

```
score = 0;
```

[keyboard press space event]:
Код:

```
score += irandom(5) * 0.456;
```

[draw event]:
Код:

```
draw_text(x, y, score / 0.456);
```

Таким образом, на экране отображается одно значение, а в памяти хранится всего 456/1000 от него. Конечно, желательно использовать более сложные формулы, чем просто деление и умножение.

Метод 2:

Можно создать ещё одну переменную для проверки счёта.
Пример реализации:
[create event]:
Код:

```
score = 0;
score_prev = score + 240.764;
```

[step event]:
Код:

```
if score != score_prev - 240.764
{
  show_message('Не пытайтесь взломать нашу игру. Это бесполезно.')
game_end();
}
```

[keyboard press space event]:
Код:

```
score += irandom(5) * 0.456;
score_prev = score + 240.764;
```

[draw event]:
Код:

```
draw_text(x, y, score);
```

Таким образом, как только пользователь изменит кол-во счёта с помощью сторонних программ - игра сразу же это обнаружит.

2.4.12. Как сделать паузу игры при открытии меню?

Ответ: Есть несколько способов это сделать. Если меню текстовое, с управлением на клавиатуре - можно использовать рекурсию. Если же меню создано объектами - есть два основных способа:
1) В первую очередь, нужно установить текущей комнате **persistent**. Перед выходом в меню нужно сохранить скриншот игры в глобальную переменную, после этого - перейти в комнату меню. В комнате меню устанавливаем как фон скриншот игры. При клике на "Продолжить" - просто возвращаемся в комнату с игрой и удаляем скриншот игры. Всё просто. Базовый код этого метода:

Код:

```
// Выход в меню:
global.game_screen = background_create_from_screen(0, 0, view_wview, view_hview, 0, 0); // Создаём скриншот экрана
room_goto(rm_menu); // Переходим в меню

// Room creation code комнаты rm_menu (settings -> Creation code):
background_index[0] = global.game_screen;
background_visible[0] = 1;

// Возвращение в игру:
background_delete(global.game_screen);
room_goto(rm_game);
```

2) Перед выходом в меню нужно сделать скриншот игры, после чего деактивировать все объекты (кроме тех, что нужны для работоспособности меню). Когда объекты деактивируются - они перестают отрисовывать свой спрайт, потому то нам и нужно запечатлить текущее состояние игры. После деактивации нужно создать кнопки и в **draw event** какого-нибудь контроллера отрисовывать скриншот игры как фон для меню (плавное затемнение и всякие эффекты для фона - это уже детали). При возвращении в игру активировать все объекты, удалить кнопки, убрать рисование скриншота игры и удалить сам скриншот. Вот базовый код этого метода:

Код:

```
// Выход в меню:
global.game_screen = sprite_create_from_screen(0, 0, view_wview, view_hview, 0, 0, 0, 0); // Создаём скриншот экрана
instance_deactivate_all(true); // Деактивируем все объекты, кроме текущего
instance_activate_object(o_menu_controll); // Активируем контроллеры, нужные для работы меню
o_menu_controll.draw_game_screen = 1; // Указываем, чтобы объект o_controll рисовал скриншот игры на фоне
instance_create(view_xview, view_yview, o_button); // Создаём кнопки

// Возвращение в игру:
sprite_delete(global.game_screen); // Удаляем скриншот игры
instance_activate_all(); // Активируем все объекты
o_menu_controll.draw_game_screen = 0; // Указываем, чтобы объект o_controll более не рисовал скриншот игры на фоне
with o_button
  instance_destroy(); // Удаляем все кнопки
```

```
// Draw event объекта o_menu_controll:
if draw_game_screen
{
  draw_sprite(global.game_screen, 0, view_xview, view_yview);
}
```

2.4.13. Как сделать респавн монстров через определённое время за границами комнаты в зависимости от счета игрока?

Ответ: Чтобы сделать респавн монстров через определённое время, можно использовать таймеры. Для начала нужно в **create event** объекта, который будет создавать монстров, написать такой код:

Код:

```
alarm[0] = room_speed; // Монстры появятся через 1 секунду
```

В событии **alarm 0** пишем такой код:

Код:

```
var xx, yy;
repeat score*0.5 + 10
{
  switch irandom(3)
  {
    case 0: xx = -32; yy = random(room_height); break;
    case 1: xx = room_width + 32; yy = random(room_height); break;
    case 2: xx = random(room_width); yy = -32; break;
    case 3: xx = random(room_width); yy = room_height + 32; break;
  }
  instance_create(xx, yy, o_enemy);
}
alarm[0] = 5 * room_speed; // Следующая волна монстров будет через пять секунд
```

2.4.14. Как сделать разброс пуль?

Ответ: Всё очень просто, следует только использовать **random_range** для некоторой случайной неточности. Например, код создания пули в игроке можно составить таким образом:

Код:

```
var bull;
bull = instance_create(x, y, o_bullet); // Создаём пулю
bull.direction = direction + random_range(-20, 20); // Максимальное отклонение - 20 градусов
```

2.4.15. Вопрос 1: Как проверить, чётное ли число? Вопрос 2: Как проверить, кратно ли одно число другому?

Ответ: Если проверка **value mod 2** возвращает **0** - число чётное, в другом случае - число нечётное. Пример:

Код:

```
if value mod 2 == 0
{
  show_message('Число чётное.');
}
else
{
  show_message('Число нечётное.');
}
```

Для проверки кратности используется та же проверка, только вместо 2 - делитель.

2.4.16. Как сделать бесконечную комнату? Движущуюся вниз/вверх.

Ответ: Обычно эффект бесконечности достигается посредством движением фона и объектов - движется фон, а игроку кажется, что движутся объекты. Предположим, что фон двигается со скоростью -5 - каждый шаг двигается на 5 пикселей вверх. Тогда недвижимый объект на экране падает со скорость 5 пикселей, так как относительно фона каждый шаг он сдвигается на 5 пикселей вниз. Объект, который движется вверх со скоростью 5 пикселей - завис в воздухе, так как относительно фона он не сдвигается. Таким образом, создавая за нижней границей комнаты объекты и устанавливая им отрицательную вертикальную скорость - получим бесконечный поток объектов. Не стоит забывать, что все объекты нужно удалять, если они пересекли верхнюю границу комнаты.

2.4.17. Как сделать, чтобы при переходе в другую комнату в текущей сохранялись все изменения?

Ответ: Нужно установить **persistent** в настройках комнаты, вкладка **settings**. Так же для этого можно использовать функцию **room_set_persistent** и переменную **room_persistent**.

2.4.18. Как загрузить спрайт/звук/фон из папки, как загрузить анимацию, как проверить существование файла?

Ответ: Для загрузки спрайтов используются следующие функции: **sprite_add** (загружает новый спрайт в игру), **sprite_replace** (заменяет существующий спрайт), **sprite_add_sprite** (добавляет спрайт в формате **gmspr**) и **sprite_replace_sprite** (заменяет существующий спрайт спрайтом в формате **gmspr**). Аналогичные функции для загрузки фонов: **background_add**, **background_replace**, **background_add_background**, **background_replace_background**. И функции для загрузки звуков: **sound_add** (добавляет музыку/звук), **sound_replace** (заменяет музыку/звук).

Чтобы сохранить анимацию в **png** формате - спрайт нужно сохранить как стрип: **Edit Sprite -> File -> Save as PNG File** (Game Maker автоматически объединит все кадры).
Чтобы загрузить полученную картинку в игру - в аргументе **imgnumb**, функции для загрузки спрайта нужно указать кол-во кадров анимации. В редакторе изображений GM загружать стрип следует выбрав пункт «**Create from Strip**» в меню «**File**».
Советую проверять на существование каждый внешний файл. Для этого используется функция **file_exists**.
Так же обязательно убедитесь, что вы не загружаете один и тот же ресурс более одного раза, и так же примите во внимание, что загруженные ресурсы не сохраняются при вызове функции **game_save**.

2.4.19. Как узнать путь к папке windows, appdata и т.д.?

Ответ: Для этого следует использовать Переменные среды / Environment variables.
Пример:
Код:

```
AppDataDir = environment_get_variable('AppData');
```

2.4.20. Вопрос 1: Как сделать, чтобы при столкновении отнимались не сразу все три жизни, а только одна? Вопрос 2: Как сделать, чтобы событие столкновения происходило через определённые промежутки времени?

Ответ: Чтобы жизни отнимались не сразу все три, а только через определённые промежутки времени - достаточно установить таймер, который идёт во время столкновения и сбрасывается на 0, если столкновения нет. К примеру, можно написать такой код в **step event** противника::
Код:

```
var obj;
obj = instance_place(x, y, o_player);
if obj != noone // Если противник столкнулся с игроком
{
  if alarm[0] == -1 // Если таймер не запущен
  {
     with obj
        hp -= 1; // Уменьшаем кол-во жизней игрока
     alarm[0] = 15; // Через 1/2 секунды противник сможет атаковать ещё раз.
  }
}
else
{
  alarm[0] = -1; // Если столкновения нет - сбрасываем таймер
}
```

alarm 0 противника:
Код:

```
// Противник может атаковать
```

Ещё один вариант - после атаки делать игрока на время бессмертным. Тогда код примет примерно такой вид:
step event противника:

Код:

```
var obj;
obj = instance_place(x, y, o_player);
if obj != noone
{
  if obj.alarm[0] == -1
  {
    with obj
    {
      hp -= 1;
      alarm[0] = 15; // Через 1/2 секунды игрок перестанет быть бессмертным.
    }
  }
}
```

alarm 0 event игрока:

Код:

```
// Игрока можно атаковать
```

2.4.21. Как определить, в какой ячейке произвольной сетки расположена конкретная позиция?

Ответ: Предположим, что сетка начинается с координат **start_x**, **start_y**. Ширина и высота сетки в ячейках - **grid_width** и **grid_height** соответственно. Ширина и высота ячейки в пикселях - **cell_width**, **cell_height**. Данная позиция - **xx**, **yy**. Итак, получаем такой код:

Код:

```
if xx >= start_x and yy >= start_y and xx <= start_x + grid_width*cell_width and yy <= start_y +
grid_height*cell_height
{
  cell_x = (xx - start_x) div cell_width;
  cell_y = (yy - start_y) div cell_height;
}
else
{
  cell_x = -1;
  cell_y = -1;
}
```

К примеру, если сетка начинается в координатах **0,0** и она никак не ограничена, в том

числе номера её ячеек могут принимать отрицательные значения, то поиск ячейки, на которой расположена позиция **xx**,**yy** будет выглядеть так:

Код:

```
cell_x = xx div cell_width;
cell_y = yy div cell_height;
```

2.4.22. Как сделать у противников разные жизни? Я пишу для одного противника health = 0, а умирают все.

Ответ: health - глобальная переменная, обычно используемая для кол-ва жизней игрока. Чтобы у каждого противника были свои жизни - следует использовать локальные переменные. У каждого противника в **create event** нужно объявить локальную переменную **hp**:

Код:

```
hp = 100; // Изначальное кол-во жизней - 100
```

При столкновении противника с пулей, к примеру, можно написать примерно такой код:

Код:

```
hp -= 15; // Уменьшаем hp противника
with other
  instance_destroy(); // Удаляем пулю
if hp <= 0 // Если жизни противника меньше или равны нулю, то...
{
  instance_destroy(); // Удаляем противника
  // Создаём кровь и т.д.
}
```

Иногда **hp** противника уменьшается не только в событиях столкновения, но и как результат различных бонусов и т.д. Чтобы не писать код уничтожения противника во всех местах, где **hp** противника уменьшается - можно поставить такой код в **step** противника:

Код:

```
if hp <= 0 // Если жизни противника меньше или равны нулю, то...
{
  instance_destroy(); // Удаляем противника
  // Создаём кровь и т.д.
}
```

2.4.23. Как сначала проиграть анимацию до конца, а затем запустить наоборот?

Ответ: В **step event** объекта можно поставить приблизительно такой код:

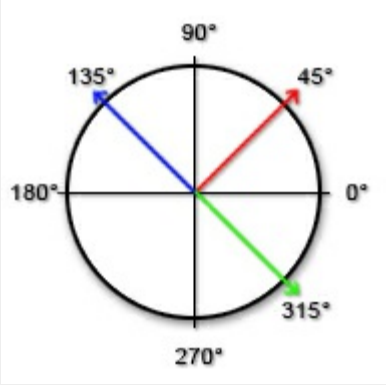
Код:

```
if round(image_index + image_speed) >= image_number // Если анимация дошла до конца, то...
  image_speed = - 1/room_speed; // Устанавливаем отрицательную скорость анимации
if image_index < 0 // Если анимация в обратную сторону закончилась, то...
  instance_destroy(); // Удаляем объект
```

2.4.24. Как вычислить разницу между направлениями?

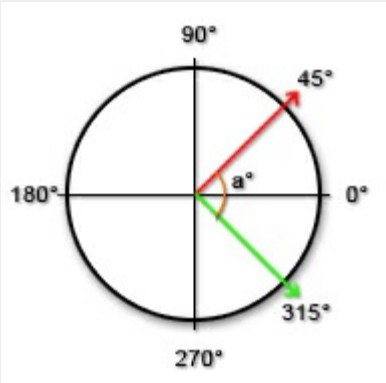
Ответ: Пожалуй, стоит разобраться с этим на примере, шаг за шагом разрабатывая искомый скрипт.

1. Предположим, у нас есть три направления:



Очевидно, что для вычисления разницы между синим и красным направлениями нужно от синего отнять красное: (135° - 45°) = 90°. Как может показаться вначале, разница между красным и синим направлениями равна разнице между синим и красным. Однако, в нашем случае, разница может принимать отрицательно значение. Поэтому, разница между красным и синим направлениями равна -90°, а не 90°. На самом деле, вы получаете значение, которое нужно добавить ко второму направлению, чтобы получить первое. Добавляя 90 градусов к 45° получается 135°, ровно как и вычитая 90 градусов из 135° получается 45°.

2. Однако с зелёным и красным направлениями такая простая операция не даст желаемого результата. Скрипт должен возвращать "ближайшую разницу", не 270 градусов (которые можно получить, вычитая 45 градусов из 315°), а 90°:



Требуется вычислить угол **a°** - он обозначен оранжевым цветом на картинке. Очевидно, что его можно получить, вычитая из 360 345° (зелёное направление), и прибавив к нему 45° (красное направление):

Код: (Вычисление угла a°)

```
dir = dir1 + (360 - dir2); // 45 + (360 - 315) = 90
```

Где **dir** - угол **a°**, **dir1** - меньший угол (45° на картинке) и **dir2** - больший угол (315 градусов на картинке).
Итак, если разность направлений меньше 180 - можно просто отнять от одного направления другое, как в случае с синим и красным направлениями. Если же разность больше или равна 180 - нужно вычислять направление посредством вышеприведённой формулы, или вот так:

Код: (Алтернативный способ вычисления)

```
dir = 360 - (dir2 - dir1);
```

Этот код аналогичен предыдущему, в нём лишь переставлены значения, хотя логика сохранена.

3. Итого, скрипт вычисления разницы между двумя направлениями можно записать так:

Код: (Первая версия скрипта)

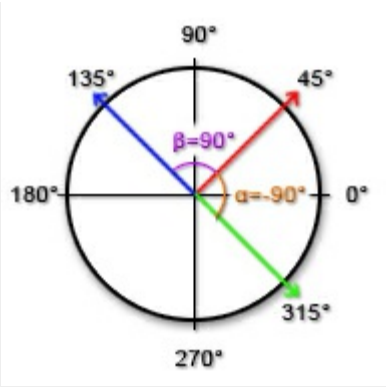
```
var diff;
diff = argument1 - argument0; // Вычисляем разность между направлениями
if abs(diff) > 180 // Если разность по модулю больше 180
  return 360 - diff; // Возвращаем разность "через ноль"
else
  return diff; // Возвращаем обычную разность
```

Мы уже вплотную приблизились к решению задачи. Однако, этот скрипт полностью рабочий только когда **argument1** больше **argument0**. К примеру, если **argument1** = 15°, а **argument0** = 215°, получим соответственно 360° - (15° - 215°) = 560 градусов. Потому, нужно изменить скрипт так, чтобы этой ошибки не возникало:

Код: (Вторая версия)

```
// argument0 - первое направление
// argument1 - второе направление
var diff;
diff = argument1 - argument0;
if abs(diff) > 180
  return (360 - abs(diff)) * -sign(diff); // Сначала находим разность между направлениями "через ноль", а потом устанавливаем нужный знак
else
  return diff;
```

Этот скрипт полностью рабочий и готов к использованию. Он возвращает значение, на которое больше или меньше направление **argument1,** чем направление **argument0**. Если **argument1** = 135, а **argument0** = 45, то получим 90. Если же **argument1** = 315, а **argument0** = 45, получим -90, так как 315 на 90 градусов меньше чем 45, считая "через ноль" (извините за каламбур, но это так):



4. Есть однако ещё одна проблема. Иногда направления бывают отрицательные, и часто превышают 360. Поэтому, желательно обработать входящие данные должным образом, чтобы не возникало непредвиденных ошибок:

Код: (Ограничение от -360 до 360)

```
dir = dir mod 360;
```

Эта строка вычисляет остаток от деления указанного направления на 360. То есть, если направление 370, то остаток от деления на 360 будет 10, если направление -740, то получим -20 и т.п. Таким образом, направление будет всегда меньше 360. После этого отрицательное направление нужно преобразовать в положительное. Сделать это, очевидно, можно так:

Код: (Преобразование направления в положительное)

```
if dir < 0
  dir += 360;
```

5. К слову, все три вышенаписаные строки можно записать всего одной:

Код: (Преобразование направления к виду 0-360)

```
dir = (dir mod 360 + 360) mod 360;
```

Поясню этот код. Вначале направление приводится к виду от -360 до 360 (не включительно), с помощью оператора **mod**: «**dir mod 360**». Далее, если направление отрицательное - она преобразовывается в положительное, после прибавления к нему 360 градусов. К примеру: "-20 + 360" это тоже самое что "360 - 20", то есть, 340 - получаем нормальное положительное направление. Далее вычисляется остаток деления направление на 360, если направление было отрицательным - ничего не изменится, так как 340 **mod** 360 = 340, а если положительным - оно вернётся к изначальному: (20 + 360) **mod** 360 = 380 **mod** 360 = 20. Если вас эта сокращённая запись чем-то смущает, либо вы не можете с нею разобраться, либо же вам просто лень морочить себе голову, то используйте обычный код:

Код:

Код: (Преобразование направления к виду 0-360 (2))

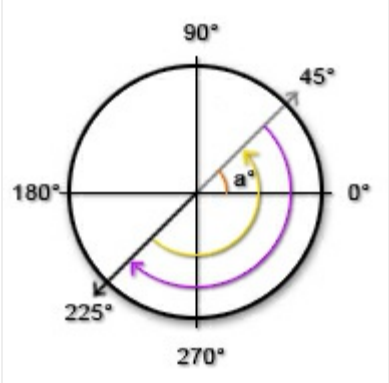
```
dir = dir mod 360;
if dir < 0
  dir += 360;
```

6. В результате всех манипуляций с направлениями, скрипт вычисления разницы между двумя направлениями приобрёл следующий вид:

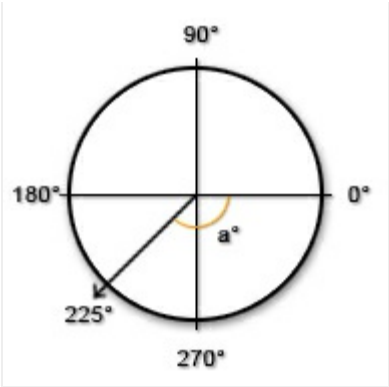
Код: (Окончательная версия скрипта)

```
/*************************************************
 Скрипт возвращает разницу между двумя направлениями.
 argument0 - первое направление;
 argument1 - второе направление направление;
*************************************************/
var diff;
diff = (argument1 - argument0) mod 360;
if diff < 0
  diff += 360;
if abs(diff) > 180
  return (360 - abs(diff)) * -sign(diff);
else
  return diff;
```

7. На этом можно было бы закончить, однако есть способ записать весь этот скрипт в одну строку, только для самых "крутых" программистов. Я хочу вам рассказать про этот способ. Идея, на самом деле, довольно простая, только очень запутанная. Для начала нужно вычислить разницу двух направлений, просто вычитая одно из другого. Далее, нужно добавить 180 к полученному направлению. Если разность была меньше 180, она так и останется меньше 360, а если больше, тогда мы получим направление большее 360. После этого находим остаток от деления на 360, и снова отнимаем 180. Когда это написано сухими словами - понять это довольно сложно, так что снова попробуем разобраться на примере:



Итак, мы "поворачиваем" наше направление (225 на рисунке) на 180 градусов, после чего вычисляем, на сколько полученное направление больше нуля (**mod** 360), в результате чего, получаем 45 градусов (серая стрелка на рисунке, **a°**). После этого вычитаем из полученного направления 180 градусов, вроде бы как "возвращая" направление назад, получая тоже, что и -(360 - **dir**) или **dir** - 360. На рисунке:



Если **dir** (разность направлений) меньше 180 - в результате всех операций получаем тоже направление, так как (**dir** + 180) **mod** 360 - 180 = **dir** (при условии, что **dir** меньше 180).
Итого, получаем вот такую страшную строку:

Код:

```
return (((argument1 - argument0) mod 360 + 360) mod 360 + 180) mod 360 - 180;
```

"**((argument1 - argument0) mod 360 + 360) mod 360**" - приводим разность направлений к виду 0 - 360, чтобы избежать ошибок.
"**(... + 180) mod 360 - 180**" - если направление меньше 180, то ничего не меняется, если больше - получаем отрицательную разность (то, что я описывал выше).
Можно заметить, что в этой строке есть пара повторяющихся действий, которые можно было бы объединить в одно:
"**((... + 360) mod 360 + 180) mod 360**" тоже что и "**(... + 540) mod 360**"
Итого, получаем такой скрипт:

Код:

```
return ((((argument1 - argument0) mod 360) + 540) mod 360) - 180;
```

и выглядит он именно так, как на сайте www.gmlscripts.com:

Код: (Альтернативная версия)

```
/*
**  usage:
**      diff = angle_difference(angle1,angle2);
**
**  given:
**      angle1    first direction in degrees, real
**      angle2    second direction in degrees, real
**
**  returns:
**      difference of the given angles in degrees, -180 to 180
**
**  GMLscripts.com
*/
{
  return ((((argument0 - argument1) mod 360) + 540) mod 360) - 180;
}
```

Вы можете использовать уже готовый скрипт в одну строку - а можете использовать

более простой скрипт для понимания, составленный нами выше.

2.5.1. Что быстрее выполняется, sqrt(sqr(x2-x1)+sqr(y2-y1)) или point_distance(x1, y1, x2, y2)?

Ответ: Встроенные функции GM выполняются быстрее самописных. **point_distance** будет работать быстрее.

2.5.2. Что быстрее выполняется, функции lengthdir_x(len, dir) и lengthdir_y(len, dir), или cos(degtorad(dir))*len и -sin(degtorad(dir))*len?

Ответ: Как и в случае с **point_distance**, **lengthdir_x** и **lengthdir_y** выполнятся быстрее.

2.5.3. Что выполняется быстрее, стандартные проверки GM в объекте (на подобии add event -> Keyboard -> <Left>), или собственные проверки в step event (вроде keyboard_check)?

Ответ: Стандартные события GM выполняются немного быстрее проверок в шаге. Вероятно, из-за того, что интерпретация кода, каким бы простым он ни был, занимает довольно много времени. То есть, использование стандартных GM событий покажет бо̲льшую производительность, чем многочисленные собственные проверки в **step event**. Однако, разница в скорости выполнения не столь велика, чтобы злоупотреблять этим.
P.S: я лично, например, чаще всего вручную пишу большинство проверок в **step event** объекта.

2.5.4. Если одно из выражений в условии ложно, будет ли проверяться остальная часть условия?

Ответ: В отличии от большинства языков программирования, в GML проверяется всё условие, вне зависимости, является ли одно из выражений ложным. То есть, условие написанное так:

Код:

```
if is_picture_grayscale(temp)
{
  if is_picture_fractal(temp)
  {
    // ...
  }
}
```

Будет выполнятся быстрее условия, записанного так:

Код:

```
if is_picture_grayscale(temp) and is_pisture_fractal(temp)
```

```
{
// ...
}
```

(названия скриптов взяты наобум)

2.5.5. Почему не стоит выносить действия в событиях step/draw в скрипты?

Ответ: Механизм вызова скриптов в GM невероятно медленный. Допустим, у нас есть два куска кода:

Код:

```
a = b;
```

и

Код:

```
script0();
```

script0 такого содержания:

Код:

```
a = b;
```

Первый код, повторенный 100 раз выполнялся 0.044 миллисекунды, второй (скрипт) - 0.341 миллисекунды (то есть, примерно в семь раз дольше!).
Первый код, повторенный 1000000 раз выполнялся 5401 миллисекунду, второй (скрипт) - 9864 миллисекунды.

Очевидно, что вызов скрипта занимает довольно много времени. На деле, при большом количестве использования скриптов в постоянно повторяющихся событиях (как **step** и **draw**) - **fps** значительно падает.

2.6.1. Как сделать, чтобы противники двигались к игроку? (вид сверху)

Ответ: Заставить противников двигаться к игроку есть множество способов. Если противники должны просто следовать за игроком по прямой, можно использовать следующий код:

Код:

```
if instance_exists(объект_игрока)
{
  if point_distance(x, y, объект_игрока.x, объект_игрока.y) > скорость_движения
  {
     move_towards_point(объект_игрока.x, объект_игрока.y, скорость_движения); // Двигаемся к игроку
  }
  else
  {
     x = объект_игрока.x;
     y = объект_игрока.y;
  }
}
```

Условие **if instance_exists(объект_игрока)** нужно для того, чтобы не появлялась ошибка, когда объекта игрока почему-то нет (например, когда он был удалён после смерти). Проверка **point_distance** требуется для того, чтобы противник не мог пройти позицию цели. К примеру, если скорость противника - 10 пикселей, а цель находится от него на расстоянии в два пикселя, то он попросту проскочит её, сдвинувшись на 10 пикселей в направлении цели. С проверкой он сдвинется точно к позиции цели, не перескакивая её.

Если противники должны двигаться к игроку, огибая стены, можно использовать следующий код:

Код:

```
if instance_exists(объект_игрока)
{
  if point_distance(x, y, объект_игрока.x, объект_игрока.y) > скорость_движения
  {
     mp_potential_step(объект_игрока.x, объект_игрока.y, скорость_движения, checkall);
  }
  else
  {
     x = объект_игрока.x;
     y = объект_игрока.y;
  }
}
```

О значении последнего аргумента (**checkall**) читайте в справке.

2.6.2. Зарезервировано.
2.6.3. Зарезервировано.

2.6.4. Зарезервировано.
2.6.5. Зарезервировано.
2.6.6. Зарезервировано.

2.6.7. Как сделать, чтобы противники "видели" игрока только на определённом расстоянии?

Ответ: Для этого нужно проверить расстояние между игроком и противником, делается это с помощью функции **point_distance**:

Код:

```
if instance_exists(объект_игрока)
{
  if point_distance(x, y, объект_игрока.x, объект_игрока.y) < радиус_зрения_противника
  {
    // Действия противника, когда игрок попал в поле зрения.
  }
}
```

О необходимости проверки **instance_exists** читайте в предыдущем вопросе.

2.6.8. Как сделать, чтобы противники видели игрока только под определённым углом?

Ответ: Используйте скрипт для вычисления разницы между двумя направлениями, создание которого подробно описывается в вопросе **2.4.24**. Если разница между направлением взгляда противника и направлением от него к игроку превышает половину от угла обзора противника, значит противник не видит игрока. К примеру, если противник имеет угол обзора в 120 градусов, то разница между направлением его взгляда и направлением от него к игроку должна по модулю не превышать 60 градусов (это значит, что противник видит игрока). В коде это выглядит примерно так:

Код:

```
var dir;
dir = point_direction(x, y, объект_игрока.x, объект_игрока.y);
if abs(dir_difference(image_angle, dir)) <= угол_обзора / 2
{
  // Действия противника, если игрок находится в его поле зрения.
}
```

2.6.9. Как сделать, чтобы противники не видели игрока через стены?

Ответ: Для этого нужно проверить, есть ли между противником и игроком стена. Используйте функцию **collision_line**:

Код:

```
if instance_exists(объект_игрока) // Если объект игрока существует
{
  if !collision_line(x, y, объект_игрока.x, объект_игрока.y, объект_стены, 0, 0)
  {
```

```
    // Действия противника, когда между ним и игроком нет стены.
  }
}
```

Восклицательный знак перед функцией - это знак отрицания, то есть, он сменяет **true** на **false** и **false** на **true**. Иначе говоря, без него проверка на русском языке звучала бы как "Если между игроком и противником *есть* стена", а с ним: "Если между игроком и противником *нет* стены".

2.6.10. Как сделать, чтобы противник двигался за ближайшим союзным персонажем игрока, или к самому игроку (если он ближе всех)?

Ответ: Так как объектов союзных персонажей может быть много, нам нужно обращаться к какому-то одному объекту, а не к каждому по отдельности. Для этого нужно использовать так называемых объектов-родителей (parent). Для начала нужно создать новый объект, родитель для всех союзных персонажей игрока и для него самого, после чего назначить его родителем всем нужным объектам. При обращении к родителю мы будем обращаться ко всем его дочерним объектам - ко всем союзным персонажам и игроку одновременно.
Для определения ближайшего объекта можно использовать функцию **instance_nearest**. Если ни одного экземпляра искомого объекта в комнате нет, функция возвратит **noone**, поэтому функцию **instance_exists** можно не использовать. Код будет выглядеть примерно так:

Код:

```
var obj;
obj = instance_nearest(x, y, родительский_объект); // Определяем ближайшего персонажа
if obj != noone // Если существует хотя бы один экземпляр объекта родительский_объект (или дочернего объекта), то...
{
  /// Двигаемся к нему:
  if point_distance(x, y, obj.x, obj.y) > скорость_движения
  {
    mp_potential_step(obj.x, obj.y, скорость_движения, checkall);
  }
  else
  {
    x = obj.x;
    y = obj.y;
  }
}
```

2.6.11. Как сделать, чтобы башня стреляла в ближайшего противника (речь о жанре "Башенная защита")?

Ответ: Код будет практически аналогичен тому, который представлен в предыдущем вопросе:

Код:

```
var obj;
obj = instance_nearest(x, y, объект_противника); // Определяем ближайшего противника.
if obj != noone // Если существует хотя бы один противник, то...
{
   var bull;
   bull = instance_create(x, y, объект_пули); // Создаём пулю.
   with bull
   {
      move_towards_point(obj.x, obj.y, скорость_пули); // Направляем её к противнику.
   }
}
```

Примеры

Платформеры:

- motion in platformer.gmk (10.6) - базовый пример двжения в платформере.
- rough surface.gmk (13.5 кб) - пример движения по неровной поверхности.
- stairs.gmk (12.2 кб) - пример лесниц. Автор: **Лер**.

Другое:

- momemtary bullets.gmk (11.6 кб) - пример мгновенных пуль, собственных жизней у противников и хелсбаров.
- healthbar and gradual turn.gmk (14.4 кб) - пример спрайтового хелбара и плавного поворота.
- depth example.gmk (25.2 кб) - пример определения наивысшего объекта в области мышки.
- tds move.gmk (9.83 кб) - пример движение по диагонали с той же скорость, что и обычно.

Ошибки, выдаваемые интерпретатором

Тема на hellroom.ru

Часть текста: Hummer. Ещё часть взята с форума yoyo, перевод Dmi7ry. Также спасибо FanTom.

Сообщения о критических ошибках

Array index>=32000

Индекс массива больше или равен 32000
Это сообщение говорит вам, что вы пытаетесь использовать индекс массива больше или равный 32000. В Game Maker максимальный размер массива может быть 32000. Это означает, что массив будет иметь максимум 32000 элементов, и индекс последнего элемента будет 31999.

Cannot find the file

Не найден файл
Это случается, когда ваш Game Maker файл заменен, перемещен или удален. Решение: найдите файл Game Maker самостоятельно.

Division by zero

Деление на ноль
Это сообщение появится, когда вы пытаетесь разделить одно число на 0. Способ избежать этой ошибки - делать проверку, прежде чем делить:

Код:

```
if a!=0 { my_variable=1/a }
```

Error Creating Stand-Alone: Cannot find the required dlls

Это происходит, когда файл **dxdata** поврежден или удален. Решение: переустановить Game Maker.

Error Creating Stand-Alone: Cannot find valid runner data

Это происходит, когда файл **rundata** поврежден или удален. Решение: переустановить Game Maker.

Error Creating Stand-Alone: Cannot save the game data

Обычно эта ошибка возникает, если игра (откомпилированный EXE) уже запущена, и при этом попытатьься ее перекомпилировать. Но, так как EXE запущен, он защищен от записи в это время.

No action libraries have been found

Не найдены библиотеки действий
Это происходит, когда библиотеки действий заменены или удалены. Библиотеки действий, входящие в Game

Maker: 02_main1, 03_main2, 07_draw, 06_extra, 05_score, 04_control, 01_move. Решение: Поместить их в каталог Lib или переустановить Game Maker.

Сообщения об ошибках компиляции

Assignment operator expected

I. Ожидается оператор присваивания
Происходит, когда вы используете имя переменной без оператора присваивания:
Код:

```
my_variable;//ОШИБКА
```

Это объявление переменной не имеет никакого смысла, поэтому вы получите сообщение об ошибке. Еще один способ получить это сообщение об ошибке - забыть поставить скобки после функции:
Код:

```
instance_create;
```

Чтобы решить эту проблему, просто добавьте после функции скобки, и, если требуется, аргументы этой функции:
Код:

```
instance_create(x,y,my_object);
```

II. Ожидается оператор присваивания

Попросту говоря, нет знака равно или он поставлен больше, чем 1 раз (исключение - оператор if)
Код:

```
g+p
t/qwerty
a+100500
image_angle
```

Поставьте знак = там, где у вас идут действия (арифметические) и там, где идут функции (variable)
Код:

```
g+=p
t/=qwerty
a+=100500
image_angle=30
```

В имени переменной не должно быть знаков препинания (включая *пробел*), и оно не должно начинаться с цифры или знака препинания (! , . ; ").

Error defining an external function

Ошибка определения внешней функции.

Эти ошибки связаны с внедрением в ГМ dll библиотек. Более подробней здесь

Failed to compile scripts

Ошибка компиляции скриптов
Происходит, если есть ошибка в одном из ваших скриптов и у вас отключена опция "Показывать сообщения об ошибках" в глобальных настройках игры. Чтобы решить эту проблему, перейдите в общие настройки игры, включите опцию и запустите игру, чтобы можно было увидеть реальное сообщение об ошибке.

Failed to compile the actions in the objects

Ошибка компиляции действий в объектах
(ошибка компиляции или работы)
Происходит, если есть ошибка в вашей игре и у вас отключена опция "Показывать сообщения об ошибках" в глобальных настройках игры. Чтобы решить эту проблему, перейдите в общие настройки игры, включите опцию и запустите игру, чтобы можно было увидеть реальное сообщение об ошибке.

Game has no rooms

В игре нет комнат
Комната - место, где происходит действие игры. Таким образом, без комнат игра не может работать. Экземпляры объектов не могут быть созданы нигде, кроме комнат. Итак, без комнат игра не может выполнить никаких скриптов или действий.

Program ends before end the code

Программа кончилась прежде, чем кончился код

Game Maker считает, что если в начале кода поставлен знак { , то там, где он закроется, будет конец программы.
Пример такого безобразия:
Код:

```
{
g=5
}
s=4
```

Разберём:

- 1 строчка - знак { . По умолчанию начало программы.
- 2 строчка - присвоение переменной.
- 3 строчка - знак } . То есть данный код закончлся и Game Maker в недоумении, почему пошли ещё какие-то символы - поэтому сильно ругается.

Исправим:
Код:

```
{
g=5
s=4
}
```

Sound/Background/Image/Sprite/File does not exist

Звук/Фон/Картинка/Спрайт/Файл не существует.

Появляется, если не получается найти объект по указанному пути (например, его там нету, либо не правильно прописан путь или имя файла)
Проверьте правильность написания имени файла (с расширением) и путь, по которому этот файл должен находиться.

Symbol <symbol> Expected

Ожидается указанный символ
В этом случае Вы забыли поставить заключительный символ, например, "Символ } ожидается" или "Символ) ожидается".

Код:

```
for (i=0; i<10; i+=1
show_message('Ой');
```

Или

Код:

```
{
show_message('Это также выдаст ошибку');
```

Часто появляются во вложенных блоках, где обычно забывают поставить закрывающую скобку в правильном месте.

Unexpected error occurred when running the game

Неожиданная ошибка при исполнении игры
Это говорит Вам, что была ошибка вне Game Maker. Отсутствие достаточного объёма графической памяти или не установленный DirectX8.0 или более поздний, являются наиболее распространенными ошибками. С этой ошибкой игра может запуститься на некоторых компьютерах и не работать на других, поскольку это зависит исключительно от индивидуальных характеристик каждого компьютера. Надёжный способ получить это сообщение об ошибке во время выполнения игры состоит в том, чтобы создать бесконечные циклы или использовать слишком много памяти.

Wrong number of arguments to function or script

Неправильное количество аргументов в функции или скрипте.

- посмотрите в справке эту функцию и сравните количество аргументов
- также возможно, что вы используете старую команду (использующуюся в более ранних версиях GM)

Сообщения об ошибках во время выполнения

Cannot assign variable

Невозможно присвоить значение переменной.

В гм есть особый вид данных: константы и **read-only** переменные (переменные только для чтения). Пример констант:
true=1
false=0
pi=3.14
ev_alarm
ev_destroy
vk_enter
и т д
Они постоянны на всём протяжении существования.

Read-only переменные - в справке они помечены звёздочкой. Их нельзя задать кодом. К примеру
Код:

```
room_width
room_height
fps
current_time
и т д
```

Раз они не могут изменяться, то такая запись как
Код:

```
room_height=500
```

и приведёт к этой ошибке. Будьте внимательны!

Cannot compare arguments

Невозможно сравнить аргументы
Происходит, когда Вы пытаетесь сравнить строку с числом. Используйте функции **real()** или **string()**, чтобы преобразовать один из аргументов в правильный формат.

Creating istance for non-existing object

Создание экземпляра несуществующего объекта
Наиболее распространенный источник этой ошибки - использование функции [b]instance_create(x,y,object)[/b] с именем объекта, которое было неправильно написано. Проверьте, что название объекта написано правильно.

File is not opened for reading

Файл не открыт для чтения
Эта ошибка появляется, если попытаться прочитать данные из файла или записать данные в файл, к которому не был получен доступ. Чтобы открыть файл для того, чтобы читать, Вы должны использовать функцию **file_text_open_read("path_to_the_file/name_of_the_file.exe")**.
Наиболее распространенная ошибка - указать неправильное имя к пути файла, или использовать абсолютные пути. Абсолютные пути, такие как **"C:\My_directory\Myfile.txt"** будут работать в Вашем компьютере, но потерпит неудачу в компьютерах других людей, если у них не будет точно такого же пути к файлу (например, диск **D:** вместо **C:**). Вместо того, чтобы использовать абсолютные пути, используйте пути относительно директории вашей игры.

INI files must be located in the same directory as the program

INI файлы должны располагаться в той же директории, что и игра
Это происходит, когда Вы пытаетесь использовать файл INI в своей игре, но Вы поместили его вне каталога игры. Переместите файл INI, который Вы пытаетесь открыть в каталог игры, или используйте другой тип файла (например, TXT).

Moving to next room after the last room, либо Moving to previous room after the first room

Попытка перемещения в следующую комнату после последней
Попытка перемещения в предыдущую комнату из первой

Не существует следующей (или предыдущей) комнаты, в которую вы собрались перемещаться.
Если у вас разработка игры, то просто сделайте новые комнаты, иначе поставьте ограничение

Код:

```
if(room_exists(следующая комната))
{
room_goto_next()
}
```

Negative array index

Отрицательный индекс массива
Это означает, что вы пытаетесь использовать отрицательный индекс в массиве. Первый элемент массива всегда элемент с индексом 0. Нет ничего перед ним, поэтому используя отрицательный индекс, вы пытаетесь обратиться к несуществующему элементу.

Only 1- and 2-dimensional arrays are supported

Поддерживаются только одномерные и двухмерные массивы.

Например **j[1]=5** или **j[7, 2]=5**

Unexpected symbol in expression

Неожиданный символ в выражении.

- вы использовали в названии переменной русские буквы - переименуйте её
- вы использовали русское имя ресурса, который написали в коде - переименуйте ресурс
- вы поставили лишнюю запятую или скобку

  Код:

  ```
  instance_create(100, 500,, hero))
  ```

- вы поставили знак равно, но ничего не присвоили
- вы не поставили точку с запятой после объявления переменных типа var и globalvar

Unknown variable or array index out of bounds

Неизвестная переменная или индекс не соответствует размерам массива.

Эта ошибка возникает в том случае, если индекс элемента, к которому обращается пользователь, меньше нуля или превышает максимальное количество элементов в массиве.
Перед тем, как обращаться к элементам массива, их нужно сначала определить (инициализировать):
Код:

```
mas[0] = 1;
mas[1] = 2;
...
```

Так же не стоит забывать, что при объявлении массива указывается общее количество элементов массива, но при обращении к массиву нумерация элементов начинается с нуля (то есть максимальное значение номера аргумента должно быть на единичку меньше, чем размер массива)

Unknown function or script 'name of function'

Неизвестная функция или скрипт *'имя функции'*
I. Происходит, когда Вы пишете с орфографическими ошибками имя функции или название скрипта, или когда Вы вызываете скрипт, который не существует. Проверьте правописание.

II.

- вы сделали ошибку в названии функции или скрипта (встроенного или самописного)
- возможно вы использовали переменную как функцию. Например:
  Код:

```
image_angle(30) // Ошибка. Правильно image_angle=30
sprite_index(s_hero) // Ошибка. Правильно sprite_index=s_hero
```

Unknown variable 'variable_name'

Неизвестная переменная *'имя переменной'*
I.Это сообщение об ошибке появляется, когда Вы пытаетесь использовать переменную, которая не объявлялась. Перед использованием переменной в любом случае, Вы должны объявить её, присваивая ей значение. Вы можете также проверить, что опция "Обращаться с не инициализированными переменными как с 0" в меню опций игры. Тогда Game Maker автоматически будет присваивать все переменные значение 0, когда они используются в первый раз.

II.

- Возможно, что вы просто опечатались при наборе имени переменной
- Возможно вы не объявили переменную в событии Create. Например в событии Step вы пишете
  Код:

```
get+=5
```

  Но ГМ не знает чему get было равно изначально, поэтому ошибка.

- Возможно вы используете переменную, определённую другим объектом. Чтобы это исправить используете
  Код:

```
global.+имя переменной

или
```

```
(объект у которого определена переменная).имя переменной  //например hero.hp

или
with(объект у которого определена переменная)
{
//действия с переменной
}
```

- Отсутствует объект, который содержит эту переменную, к которой вы обращаетесь.
  Например, это может происходить, когда противник движется по направлению к игроку:
  Код:

```
move_towards_point(Player.x,Player.y,3);
```

  Когда игрок погибает, появляется ошибка. Чтобы ее избежать, используйте условие:
  Код:

```
if instance_exists(Player) {move_...}
```

Variable name expected

Ожидается имя переменной
I. Обычно происходит, когда Вы пытаетесь создать переменную с тем же самым именем как один из Ваших ресурсов (звук, спрайт, объект, шрифт, и т.д.) или когда имя переменной начинается с цифры вместо буквы. Лучшим способом избежать этого - добавлять к имени ресурса приставку, указывающую тип ресурса.
Например:
obj_myobject (или **o_myobject**)
spr_mysprite (или **s_mysprite**)
fnt_myfont (или **f_myfont**)

II. Это значит, что что-то не так с именем переменной:

- имя переменной совпадает с именем любого ресурса
- нет имени переменной, когда функция возвращает какой-либо индекс

Исправить можно следующими способами:
1) Поменять имя переменной, чтобы не совпадало с именем ресурса или поменять название ресурса.
2) Присвоить функции переменную
Код:

```
переменная=имя_функции()
```

Wrong type of arguments to +

I. Неверный тип аргумента для сложения
Это происходит, когда Вы пытаетесь добавить число к строке, или наоборот. Пример:
Код:

```
value=1;
show_message("1"+value);
```

Чтобы исправить это, Вы должны преобразовать число в строку, используя функцию [b]string()[/b]:

Код:

```
value=1;
show_message("1"+string(value));
```

II. Неверный тип аргумента для операций + - * /

100% вы попытались произвести арифметическую операцию со значениями разных типов (строковые и числовые)
Воспользуйтесь переводом из одного вида в другой
Если вам нужно склеить строки, то используйте
Код:

```
string(число)
```

Во всех остальных случаях переводите в числа
Код:

```
real(строка)
```

Учтите, что вычитать, умножать и делить строковые значения нельзя.

Wrong type of array index

Не подходящий тип индекса массива.

Связанно, опять же, с массивами.
Индекс массива поддерживает только числовые значение.
Разрешённые варианты записей:
Код:

```
Правильно:
array[2]='123'
i=7
ar[i]=1

Ошибка:
array['2']='123'
i='7'
ar[i]=1
```

Для перевода строковых значений в числовые используйте функцию real()

Словарь терминов и сокращений

Перед вами небольшой список сокращений, сленговых выражений и терминов, связанных с разработкой игр. Тема на gmakers.ru
Авторы: DeatHSoul, Hummer, drdan1959, sashok_one, Edikoz, N_hound, Макасин и Ogion.
Также были использованы материалы из свободной энциклопедии Wikipedia.

Термин	Варианты	Описание
Alarm	аларм, будильник, таймер	событие, происходящее через определенное количество шагов (времени)
API	АПИ	(англ. Application Programming Interface) — набор готовых классов, процедур, функций, структур и констант, предоставляемых приложением (библиотекой, сервисом) для использования во внешних программных продуктах. К примеру, API социальной сети Вконтакте позволяет использовать данные из сети в приложениях и социальных играх.
GM	Гамак, ГМ	Game Maker.
GML код		написанный человеком текст компьютерной программы на яп GML. Подробнее
GUI	ГУИ, ГУЙ, ГИП, ГПИ	(англ. Graphical User Interface) графический пользовательский интерфейс. Это может быть как набор элементов интерфейса программы, так и непосредственно интерфейс игры. Подробнее
Hot Seat		игра по очереди на одном компьютере.
LMB(ЛКМ)	RMB(ПКМ), MMB(СКМ)	левая, правая и средняя кнопки мыши соответственно.
PS	ЗЫ	(лат. post scriptum — «после написанного») — дополнение к тексту сообщения. Возможны также модификации, вроде P.P.S. (post post scriptum), P.P.P.S. (post post post scriptum) и т.п.
Splitscreen		разделение экрана на несколько частей, каждый из игроков играет на своей части.
Surface	Сурфейс	Поверхность. Подробнее
A*	А звезда, А стар (англ. star – звезда)	алгоритм поиска пути.Подробнее
АртМани	ArtMoney	популярная читерская программа, позволяющая изменить любую переменную (жизни, деньги и т.п.) в любой игре. Подробнее
Ачивмент	ачивка	(англ. achievement - достижение, успех) специальный бонус в игре, выдаваемый за выполнение необязательного, чаще всего сложного задания.
Баг	Bug	ошибка в игре. Подробнее
БГ	бэк,Back,задник	(англ. BackGround) фон.
Булевая переменная	Boolean	переменная, которая может принимать только два значения: истина (true,1) и ложь (false,0). Подробнее
ГГ		главный герой.
Геймдев	gamedev, Геймдевелопинг, gamedeveloping	(англ. game, developing) разработка игр.
Геймдевелопер	gamedeveloper, Геймдевелопер	(англ. game, developer) разработчик игр.
Геймплей	gameplay	игровой процесс в компьютерной игре. Подробнее
Глюк		ошибка в игре, чаще всего зависящая от характеристик компьютера. Обычно игроки к глюкам игроки так же причисляют всевозможные ошибки, зависания — любую некорректную работу игры. Часто глюки не постоянны, и перезапуск игры может исправить ситуацию.
Движок	(англ. engine – мотор, двигатель)	часть программы, отвечающая за выполнение прикладных задач, например: поиск, отображение графики, проигрывание звука, обработка

		физики и т.п. Подробнее
Девелопер	(англ. developer)	разработчик.
Диздок	(англ. Design document, GDD)	дизайн-документ, максимально полное описание игры. Он позволяет разработчикам игры составить «план дальнейших действий» по воплощению задуманного проекта в проект реальный. Подробнее
Зависание		состояние, вызываемое багом или глюком, при котором игра или даже вся ОС не отвечает на действия пользователя.
ИМХО	IMHO	известное выражение, означающее «по моему скромному мнению» (In My Humble Opinion или In My Honest Opinion). В некоторых случаях можно расшифровать как «имею мнение, хрен оспоришь».
Интерфейс		набор методов, посредством которых происходит взаимодействие пользователя с игрой. Чаще всего подразумевается графический интерфейс - визуальное оформление игры. Подробнее
Исходник	Сорец	(англ. source) исходный файл проекта — *.gmk, *.gm6 и т.д. Иногда под исходником так же подразумевают бэкап-файлы: *.gb1 - *.gb9
Квест	(англ Quest – поиск предметов, поиск приключений)	жанр игр. Часто также употребляется в смысле «задание», которое должен выполнить игрок.
Конструктор игр		программа, позволяющая создавать игры, используя готовые шаблоны, средства управления графикой, звуком, встроенные эффекты и т.п. Подробнее
Концепт, концепт-док		концепт-документ (англ. concept document), краткое и ёмкое описание концепции (идеи) игры, то есть, максимально сжатый документ, в котором рассказывается о том, какой будет игра, чем она будет интересна и как она должна выглядеть после разработки. Подробнее
ЛС	личка	личные сообщения.
Манабар	manabar	полоса маны.
Ориджин	(англ. origin – начало координат)	центр спрайта – центральная точка спрайта, где x и y спрайта равны нулю.
ОС		операционная система.
Пост		ответ в теме, сообщение.
Редактор	Editor	программа для создания и редактирования чего-либо. Бывают текстовые, графические, видео, аудио и т.д. редакторы. Обычно под «редактором карт» подразумевают утилиту, позволяющую создавать дополнительные пользовательские карты к игре.
Репутация	«репа», карма, рейтинг	на некоторых ресурсах у пользователей есть возможность поощрять (или выражать возмущение) друг друга, увеличивая или уменьшая друг другу репутацию. Об увеличении кармы говорят: плюс, +1, респект.
Респаун	респавн (англ. Respawn – перерождение)	в компьютерных играх повторяющееся появление какого-либо объекта или персонажа игрового мира, происходящее в определённой точке (точке респауна, англ. respawn point) игрового пространства.
Респект	(англ. respect – уважение)	Служит для выражения уважения другому пользователю.
Родитель	Parent, парент	объект, свойства которого (например, события, переменные) могут быть наследованы потомками данного объекта, если в потомках эти свойства не были переопределены. При обращении к родительскому объекту также происходит обращение к потомкам этого объекта. Подробнее
Сабж	сабдж,субж,subj	(сокращение от англ. subject – предмет, тема) предмет обсуждения, тема беседы.
Сеттинг	(англ. setting)	описание игрового мира, включающее в себя место, время и настроение. Иными словами, то окружение, в котором находятся персонажи, за исключением самих персонажей и тех событий, которые произойдут во время игры. Подробнее
Скрипт		записанный сценарий, описывающий последовательность действий. В GM это процедура или функция, создаваемая программистом. Подробнее
ТЗ		техническое задание.
Тред	топик	тема на форуме.
Трейнер		сторонняя программа-взломщик, действует аналогично чит-кодам.

Фича	(от англ. feature)	особенность игры или какой-либо технологии, демонстрирующая её отличие от остальных. Подробнее
Фонт	(англ. font)	шрифт.
Хелсбар	healthbar	полоса здоровья.
Хелсы	хиты	жизни.
Холивар	(от англ. holy war – священная война)	обмен сообщениями в интернет-форумах и чатах, представляющий собой бесплодную полемику, в которой участники яростно пытаются навязать друг другу свои точки зрения. Например, доказать друг другу преимущество одной из нескольких похожих идей.
Чит		специальный код, дающий преимущество в игровом процессе игроку, который его использует. В игру внедряется разработчиками или крекерами.
Читер		игрок, использующий читы, получающий нечестное преимущество посредством использования читов, трейнеров и т.п.
Шапка (темы, сайта)		первое сообщение темы, верхняя полоса сайта и т.д.
Шейдер	shader	программа, исполняемая на графическом процессоре. Подробнее
ЯП		язык программирования.

История

Развернуть / Свернуть

```
[+]         Добавлено
[*]         Добавлен вариант ответа в существующий вопрос
[~]         Исправлены ошибки/опечатки/формулировки/удобночитаемость и т.п
[#]         Изменение оболочки
```

14.07.2011

2.1.1 [*] Ещё один вариант решения через if

2.1.5 [~] Немного изменена формулировка

Словарь [+] определение Respown

17.07.2011

[#] Добавлен поиск, для этого сейчас все ответы по умолчанию развёрнуты

[#] Добавлены кнопки [развернуть/свернуть всё]

[+] Переработан раздел "Словарь терминов и сокращений"

[+] Добавлен раздел "Ошибки"

04.10.2011

[+] Пополнен список ошибок